

```

VVV      VVV  MMM      MMM      SSSSSSSSSSSSS  LLL      IIIIIIIII  000000000000
VVV      VVV  MMM      MMM      SSSSSSSSSSSSS  LLL      IIIIIIIII  000000000000
VVV      VVV  MMM      MMM      SSSSSSSSSSSSS  LLL      IIIIIIIII  000000000000
VVV      VVV  MMMMMM  MMMMMM  SSS      LLL      III      000      000
VVV      VVV  MMMMMM  MMMMMM  SSS      LLL      III      000      000
VVV      VVV  MMMMMM  MMMMMM  SSS      LLL      III      000      000
VVV      VVV  MMM      MMM      SSS      LLL      III      000      000
VVV      VVV  MMM      MMM      SSS      LLL      III      000      000
VVV      VVV  MMM      MMM      SSS      LLL      III      000      000
VVV      VVV  MMM      MMM      SSS      LLL      III      000      000
VVV      VVV  MMM      MMM      SSS      LLL      III      000000000000
VVV      VVV  MMM      MMM      SSSSSSSSSSS  LLL      III      000000000000
VVV      VVV  MMM      MMM      SSSSSSSSSSS  LLL      III      000000000000
VVV      VVV  MMM      MMM      SSS      LLL      III      000      000
VVV      VVV  MMM      MMM      SSS      LLL      III      000      000
VVV      VVV  MMM      MMM      SSS      LLL      III      000      000
VVV      VVV  MMM      MMM      SSS      LLL      III      000      000
VVV      VVV  MMM      MMM      SSS      LLL      III      000      000
VVV      VVV  MMM      MMM      SSS      LLL      III      000      000
VVV      VVV  MMM      MMM      SSS      LLL      III      000      000
VVV      VVV  MMM      MMM      SSS      LLL      III      000      000
VVV      VVV  MMM      MMM      SSSSSSSSSSSSS  LLLLLLLLLLLLLLLLL  IIIIIIIII  000000000000
VVV      VVV  MMM      MMM      SSSSSSSSSSSSS  LLLLLLLLLLLLLLLLL  IIIIIIIII  000000000000
VVV      VVV  MMM      MMM      SSSSSSSSSSSSS  LLLLLLLLLLLLLLLLL  IIIIIIIII  000000000000

```

```
LL      IIIIII  BBBB BBBB  CCCCCCCC  LL      IIIIII  CCCCCCCC  AAAAAA  LL
LL      IIIIII  BBBB BBBB  CCCCCCCC  LL      IIIIII  CCCCCCCC  AAAAAA  LL
LL      II      BB      BB  CC      CC  LL      II      CC      CC  LL
LL      II      BB      BB  CC      CC  LL      II      CC      CC  LL
LL      II      BB      BB  CC      CC  LL      II      CC      CC  LL
LL      II      BB      BB  CC      CC  LL      II      CC      CC  LL
LL      II      BB      BB  CC      CC  LL      II      CC      CC  LL
LL      II      BB      BB  CC      CC  LL      II      CC      CC  LL
LL      II      BB      BB  CC      CC  LL      II      CC      CC  LL
LL      II      BB      BB  CC      CC  LL      II      CC      CC  LL
LLLLLLLL  IIIIII  BBBB BBBB  CCCCCCCC  LLLLLLLLLL  IIIIII  CCCCCCCC  AAAAAA  LLLLLLLLLL  ....
LLLLLLLL  IIIIII  BBBB BBBB  CCCCCCCC  LLLLLLLLLL  IIIIII  CCCCCCCC  AAAAAA  LLLLLLLLLL  ....
```

```
LL      IIIIII  SSSSSSSS
LL      IIIIII  SSSSSSSS
LL      II      SS
LL      II      SS
LL      II      SS
LL      II      SS
LL      II      SSSSSS
LL      II      SSSSSS
LL      II      SS
LL      II      SS
LL      II      SS
LLLLLLLL  IIIIII  SSSSSSSS
LLLLLLLL  IIIIII  SSSSSSSS
```

```
0001 0 XTITLE 'LIB$CLI_CALLBACK - CLI Callback Interface Procedures'
0002 0 MODULE LIB$CLI_CALLBACK ( ! CLI Callback Procedures
0003 0 IDENT = 'V04-000' ! File: LIBCLICAL.B32 Edit: STAN3009
0004 0 ) =
0005 1 BEGIN
0006 1
0007 1 *****
0008 1 *
0009 1 * COPYRIGHT (c) 1978, 1980, 1982, 1984 BY
0010 1 * DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.
0011 1 * ALL RIGHTS RESERVED.
0012 1 *
0013 1 * THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED
0014 1 * ONLY IN ACCORDANCE WITH THE TERMS OF SUCH LICENSE AND WITH THE
0015 1 * INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR ANY OTHER
0016 1 * COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY
0017 1 * OTHER PERSON. NO TITLE TO AND OWNERSHIP OF THE SOFTWARE IS HEREBY
0018 1 * TRANSFERRED.
0019 1 *
0020 1 * THE INFORMATION IN THIS SOFTWARE IS SUBJECT TO CHANGE WITHOUT NOTICE
0021 1 * AND SHOULD NOT BE CONSTRUED AS A COMMITMENT BY DIGITAL EQUIPMENT
0022 1 * CORPORATION.
0023 1 *
0024 1 * DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE OR RELIABILITY OF ITS
0025 1 * SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.
0026 1 *
0027 1 *****
0028 1
0029 1
0030 1
0031 1 ++
0032 1 FACILITY: General Utility Library
0033 1
0034 1 ABSTRACT:
0035 1
0036 1 This module contains callable procedures which allow user programs
0037 1 to access the CLI callback facility. The procedures in this module
0038 1 are:
0039 1 LIB$GET_SYMBOL get value of a CLI symbol
0040 1 LIB$SET_SYMBOL set value of a CLI symbol
0041 1 LIB$DELETE_SYMBOL delete a CLI symbol
0042 1 LIB$SET_LOGICAL set value of a supervisor mode
0043 1 logical name
0044 1 LIB$DELETE_LOGICAL delete a supervisor mode logical name
0045 1 LIB$DISABLE_CTRL disable CLI out-of-band handling
0046 1 LIB$ENABLE_CTRL re-enable CLI out-of-band handling
0047 1
0048 1 ENVIRONMENT: Runs only in USER mode - AST reentrant
0049 1
0050 1 AUTHOR: Ralph O. Weber, CREATION DATE: 19-AUG-1981
0051 1
0052 1 MODIFIED BY:
0053 1
0054 1 1-001 - Original. ROW 19-AUG-1981
0055 1 1-002 - Change symbol name > 255 characters error to LIB$_INVSYMNAM.
0056 1 ROW 3-DEC-1981
0057 1 1-003 - Use CLISC_SRVDESC. Improve code. SBL 18-Dec-1981
```



```
.. 58      0058 1 1-004 - Correct all references to LIB$ANALYZE_SDESC. DGP 31-Dec-1981
.. 59      0059 1 1-005 - Correct PSECT definitions. SBL 4-Jan-1981
.. 60      0060 1 1-006 - Fix reference to CLIMSG.B32 to use SHRLIB$, not LIB$ TMH 14-Feb-1982
.. 61      0061 1 1-007 - Allow symbol name to start with $ or . SBL 3-Feb-1983
.. 62      0062 1 1-008 - Use new callbacks to allow itemlists and attributes. STAN 26-Feb-1984.
.. 63      0063 1 1-009 - Fix bug in way pass itemlist to callback. STAN 10-JUL-1984.
.. 64      0064 1 --
```

```
.. 66      0065 1 XSBTTL 'Declarations'
.. 67      0066 1
.. 68      0067 1 SWITCHES:
.. 69      0068 1
.. 70      0069 1
.. 71      0070 1 SWITCHES ADDRESSING_MODE (EXTERNAL = GENERAL, NONEXTERNAL = WORD_RELATIVE);
.. 72      0071 1
.. 73      0072 1
.. 74      0073 1 LINKAGES:
.. 75      0074 1
.. 76      0075 1
.. 77      0076 1 LINKAGE LIB$ANALYZE_SDESC JSB LINK = JSB (REGISTER=0; REGISTER=1, REGISTER=2):
.. 78      0077 1     NOTUSED (3,4,5,6,7,8,9,10,11);
.. 79      0078 1
.. 80      0079 1
.. 81      0080 1 TABLE OF CONTENTS:
.. 82      0081 1
.. 83      0082 1
.. 84      0083 1 FORWARD ROUTINE
.. 85      0084 1     LIB$GET_SYMBOL,      ! get value of a CLI symbol
.. 86      0085 1     LIB$SET_SYMBOL,      ! set value of a CLI symbol
.. 87      0086 1     LIB$SET_LOGICAL,     ! set value of a supervisor-mode
.. 88      0087 1     ! logical name
.. 89      0088 1     LIB$DELETE_LOGICAL,   ! delete a supervisor-mode logical
.. 90      0089 1     ! name
.. 91      0090 1     LIB$DISABLE_CTRL,    ! disable CLI out-of-band recognition
.. 92      0091 1     LIB$ENABLE_CTRL,     ! re-enable CLI out-of-band recognition
.. 93      0092 1     LIB$$BUILD_SYMBOL_NAME; ! build a good symbol name
.. 94      0093 1
.. 95      0094 1
.. 96      0095 1 INCLUDE FILES:
.. 97      0096 1
.. 98      0097 1
.. 99      0098 1 LIBRARY 'SYS$LIBRARY:STARLET';      ! System symbol definitions
100      0099 1
101      0100 1 REQUIRE 'SHRLIB$:CLIMSG';      ! CLIS_ symbols
102      0380 1
103      0381 1
104      0382 1 MACROS:
105      0383 1
106      0384 1     MACRO MOVEDESC builds a S-type string descriptor at _TO
107      0385 1     which points to the same data area as that pointed to by
108      0386 1     the string descriptor at _FROM. NB: since the _TO
109      0387 1     descriptor will be used only by SYS$CLI which ignores the
110      0388 1     DTYPE and CLASS fields, only the LENGTH and POINTER fields
111      0389 1     are built by MOVEDESC.
112      0390 1
113      M 0391 1     MACRO MOVEDESC (_FROM, _TO) =
114      M 0392 1     BEGIN
115      M 0393 1     REGISTER
116      M 0394 1     RET STATUS = 0;
117      M 0395 1     RET STATUS = LIB$ANALYZE_SDESC R2 ( _FROM;
118      M 0396 1     BLOCK [ _TO, DSC$_LENGTH; , BYTE],
119      M 0397 1     BLOCK [ _TO, DSC$_POINTER; , BYTE] );
120      M 0398 1     IF NOT .RET_STATUS
121      M 0399 1     THEN
122      M 0400 1     RETURN (.RET_STATUS);
```

```
123 0401 1      ENDX;
124 0402 1
125 0403 1
126 0404 1      EQUATED SYMBOLS:
127 0405 1
128 0406 1      NONE
129 0407 1
130 0408 1      FIELDS:
131 0409 1
132 0410 1      NONE
133 0411 1
134 0412 1      PSECTS:
135 0413 1
136 0414 1
137 0415 1      PSECT
138 0416 1      CODE = _LIB$CODE (READ, NOWRITE, EXECUTE, SHARE, PIC, ADDRESSING_MODE (WORD_RELATIVE)),
139 0417 1      PLIT = _LIB$CODE (READ, NOWRITE, EXECUTE, SHARE, PIC, ADDRESSING_MODE (WORD_RELATIVE)),
140 0418 1      OWN = _LIB$DATA (READ, WRITE, NOEXECUTE, NOSHARE, PIC, ADDRESSING_MODE (LONG_RELATIVE)),
141 0419 1      GLOBAL = _LIB$DATA (READ, WRITE, NOEXECUTE, NOSHARE, PIC, ADDRESSING_MODE (LONG_RELATIVE));
142 0420 1
143 0421 1
144 0422 1      OWN STORAGE:
145 0423 1
146 0424 1      NONE
147 0425 1
148 0426 1      EXTERNAL REFERENCES:
149 0427 1
150 0428 1
151 0429 1      EXTERNAL ROUTINE
152 0430 1      SYS$CLI, ! CLI Callback Routine
153 0431 1      LIB$ANALYZE_SDESC_R2 : LIB$ANALYZE_SDESC_JSB LINK,
154 0432 1      LIB$SCOPE_DXDX, ! Analyze a string descriptor
155 0433 1      LIB$GET1_DD, ! Copy strings of any type
156 0434 1      LIB$SFREET_DD; ! to any type
157 0435 1      LIB$GET1_DD, ! Get a dynamic string
158 0436 1      LIB$SFREET_DD; ! Free a dynamic string
159 0437 1
160 0438 1      EXTERNAL
161 0439 1      LIB$AB_UPCASE; ! the string upcase table
162 0440 1
163 0441 1      EXTERNAL LITERAL
164 0442 1      LIB$INVARG, ! Condition value symbols
165 0443 1      LIB$WRONUMARG, ! Invalid argument
166 0444 1      LIB$INVSYNMAM, ! Wrong number of arguments
167 0445 1      LIB$NOSUCHSYM, ! Illegal symbol name
168 0446 1      LIB$INSCLIMEM, ! No such symbol
169 0447 1      LIB$AMBSYMDEF, ! Insufficient CLI memory
170 0448 1      LIB$NOCLI, ! Ambiguous symbol defined
171 0449 1      LIB$UNECLIERR; ! No CLI present
! Unexpected CLI error
```



```
173 0450 1 %SBTTL 'LIB$GET_SYMBOL - get value of a CLI symbol'
174 0451 1 GLOBAL ROUTINE [LIB$GET_SYMBOL ( get value of a CLI symbol
175 0452 1     SYMBOL: REF BLOCK [, BYTE], symbol string descriptor
176 0453 1     RETBUF: REF BLOCK [, BYTE], return value string descriptor
177 0454 1     RETLEN: REF VECTOR [, WORD], no. bytes returned (opt.)
178 0455 1     MODE: REF VECTOR [, LONG] symbol table searched (opt.)
179 0456 1 ) =
180 0457 1
181 0458 1 ++
182 0459 1 FUNCTIONAL DESCRIPTION:
183 0460 1
184 0461 1 LIB$GET_SYMBOL gets the value of the specified CLI symbol and returns
185 0462 1 it in the specified buffer. Before the actual search, however, the
186 0463 1 specified symbol name is upcased so that it will better match symbol
187 0464 1 names formed by the CLI. The local-symbol table is scanned first for
188 0465 1 the symbol, and if no matching symbol is found there, the global-
189 0466 1 symbol table is scanned. The length of the returned value and a mode
190 0467 1 value, indicating the table in which the symbol value was found, are
191 0468 1 optionally returned when parameters to receive them are supplied in
192 0469 1 the calling sequence.
193 0470 1
194 0471 1 Numeric values are automatically translated to decimal strings before
195 0472 1 being returned.
196 0473 1
197 0474 1 The optional mode value can be one of:
198 0475 1     LIB$K_CLI_LOCAL_SYM = 1 ==> Local symbol table name
199 0476 1     LIB$K_CLI_GLOBAC_SYM = 2 ==> Global symbol table name
200 0477 1 These symbols are defined in $LIBCLIDEF.
201 0478 1
202 0479 1 CALLING SEQUENCE:
203 0480 1
204 0481 1     ret_status.wlc.v = LIB$GET_SYMBOL (symbol.rt.dx, retbuf.wt.dx
205 0482 1                                     [, retlen.ww.r [, mode.wl.r]])
206 0483 1
207 0484 1 FORMAL PARAMETERS:
208 0485 1
209 0486 1     symbol.rt.dx - A string, passed by descriptor, which contains the
210 0487 1                   symbol to be searched for. An upcased copy of this
211 0488 1                   string will actually be used for the search.
212 0489 1
213 0490 1     retbuf.wt.dx - A string, passed by descriptor, into which the value
214 0491 1                   of the symbol, if found, will be written.
215 0492 1
216 0493 1     retlen.ww.r - An optional word which, if present, will receive
217 0494 1                   the length of the returned symbol value string.
218 0495 1
219 0496 1     mode.wl.r - An optional longword which, if present, will receive a
220 0497 1                  value indicating table in which symbol was found.
221 0498 1                  Possible values are:
222 0499 1                      LIB$K_CLI_LOCAL_SYM ==> Local symbol table name
223 0500 1                      LIB$K_CLI_GLOBAC_SYM ==> Global symbol table name
224 0501 1
225 0502 1 IMPLICIT INPUTS:
226 0503 1
227 0504 1     NONE
228 0505 1
229 0506 1 IMPLICIT OUTPUTS:
```

```
230 0507 1
231 0508 1      NONE
232 0509 1
233 0510 1      COMPLETION STATUS: (or ROUTINE VALUE:)
234 0511 1
235 0512 1      $$$ NORMAL      Normal successful completion
236 0513 1      LIB$_INVARG     Invalid argument
237 0514 1      LIB$_INVSYMNAM  Invalid symbol name
238 0515 1      LIB$_NOSUCHSYM  No such symbol found
239 0516 1      LIB$_INSCLIMEM  Insufficient CLI memory
240 0517 1      LIB$_NOCLI     No CLI present to perform function
241 0518 1      LIB$_UNECLIERR  Unexpected CLI Error
242 0519 1
243 0520 1      SIDE EFFECTS:
244 0521 1
245 0522 1      WARNING:
246 0523 1      Although this procedure performs some checks on the validity of
247 0524 1      symbol names passed to it, some symbol name considered valid by
248 0525 1      this procedure will be considered invalid by DCL. Callers of this
249 0526 1      procedure are responsible for insuring that symbol names passed to
250 0527 1      this procedure contain only alphanumeric characters.
251 0528 1
252 0529 1      --
253 0530 1
254 0531 2      BEGIN
255 0532 2
256 0533 2      BUILTIN
257 0534 2      NULLPARAMETER;
258 0535 2
259 0536 2      LOCAL
260 0537 2      CLI_REQ_BLOCK : BLOCK [CLISC_SRVDESC, BYTE], ! A CLI request block
261 0538 2      DYN_STRING : BLOCK [8, BYTE], ! Descriptor for dynamic string
262 0539 2      RETURN_STATUS : BLOCK [4, BYTE]; ! A return status value
263 0540 2
264 0541 2
265 0542 2
266 0543 2      +
267 0544 2      Initialize CLI Command request block.
268 0545 2      -
269 0546 2      CH$FILL (0, CLISC_SRVDESC, CLI_REQ_BLOCK);
270 0547 2      CLI_REQ_BLOCK [CLISB_RQTYPE] = CLISK_CLISERV;
271 0548 2      CLI_REQ_BLOCK [CLISW_SERVCOD] = CLISK_GETSYM;
272 0549 2      RETURN_STATUS = LIB$$BUILD_SYMBOL_NAME( .SYMBOL, DYN_STRING,
273 0550 2      CLI_REQ_BLOCK[CLISQ_NAMDESC] );
274 0551 2      IF NOT .RETURN_STATUS THEN RETURN .RETURN_STATUS;
275 0552 2
276 0553 2      +
277 0554 2      Get CLI symbol value.
278 0555 2      -
279 0556 2      RETURN_STATUS = SYS$CLI (CLI_REQ_BLOCK);
280 0557 2
281 0558 2      +
282 0559 2      Free dynamic string used for upcased symbol name.
283 0560 2      -
284 0561 2      LIB$FREE1_DD( DYN_STRING );
285 0562 2
286 0563 2      +
```



```

287 0564 2 ! Process returned symbol value and requested optional values.
288 0565 2 !-
289 0566 IF .RETURN_STATUS
290 0567 THEN
291 0568 BEGIN
292 0569 LOCAL
293 0570 CLI_DESC : REF BLOCK [8, BYTE];
294 0571 CLI_DESC = CLI_REQ_BLOCK [CLI$VALDESC];
295 0572 RETURN_STATUS = LIB$SCOPY DXDX T.CLI_DESC, .RETBUFF);
296 0573 IF NOT NULLPARAMETER (3) AND .RETURN_STATUS
297 0574 THEN
298 0575 BEGIN
299 0576 LOCAL
300 0577 ADDRESS,
301 0578 LENGTH;
302 0579 LIB$ANALYZE_SDESC_R2 (.RETBUFF, LENGTH, ADDRESS );
303 0580 RETLEN [0] = MINU (.LENGTH, .CLI_DESC [DSC$W_LENGTH]);
304 0581 END;
305 0582 IF NOT NULLPARAMETER (4)
306 0583 THEN
307 0584 MODE [0] = .CLI_REQ_BLOCK [CLI$RQSTAT];
308 0585 END
309 0586 ELSE IF (.RETURN_STATUS [ST$SV_FAC_NO] EQL CLI$FACILITY)
310 0587 THEN
311 0588 RETURN_STATUS =
312 0589 BEGIN
313 0590 SELECTONE .RETURN_STATUS OF
314 0591 SET
315 0592 [CLI$UNDSYM] : LIB$NOSUCHSYM;
316 0593 [CLI$BUFOVF] : LIB$INSCLIMEM;
317 0594 [CLI$INVREQTYP] : LIB$NOCLI;
318 0595 [OTHERWISE] : LIB$UNECLIERR;
319 0596 TES
320 0597 END;
321 0598
322 0599 RETURN (.RETURN_STATUS);
323 0600
324 0601 1

```

! End of routine LIB\$GET_SYMBOL

.TITLE LIB\$CLI_CALLBACK LIB\$CLI_CALLBACK - CLI Callbac
k Interface Proce

.IDENT \V04-000\

.EXTRN SY\$CLI, LIB\$ANALYZE_SDESC R2
.EXTRN LIB\$SCOPY DXDX, LIB\$GET1 DD
.EXTRN LIB\$SFREET DD, LIB\$AB UPXSE
.EXTRN LIB\$INVARG, LIB\$WRORUMARG
.EXTRN LIB\$INVSYMNAM, LIB\$NOSUCHSYM
.EXTRN LIB\$INSCLIMEM, LIB\$AMBSYMDEF
.EXTRN LIB\$NOCLI, LIB\$UNECLIERR

.PSECT _LIB\$CODE, NOWRT, SHR, PIC, 2

.ENTRY LIB\$GET_SYMBOL, Save R2,R3,R4,R5
MOVAB -92(SP), SP
MOVCS #0, (SP), #0, #84, CLI_REQ_BLOCK

: 0451
: 0546

0054 8F 00 5E A4 AE 9E 00002
6E 00 2C 00006

```
; Routine Size: 213 bytes,    Routine Base: _LIB$CODE + 0000
```


LIBSCLI_CALLBAC LIBSCLI_CALLBACK - CLI Callback Interface Proce M 8
V04-000 LIBSGET_SYMBOL - get value of a CLI symbol 16-Sep-1984 02:22:35 VAX-11 B11sg-32 V4.0-742 Page 9
14-Sep-1984 13:27:47 [VMSLIB.SRC]LIBCLICAL.B32;1 (3)

LIB
VOL

```

0602 1 XSBTTL 'LIB$SET_SYMBOL - set value of a CLI symbol'
0603 1 GLOBAL ROUTINE LIB$SET_SYMBOL (
0604 1     SYMBOL: REF BLOCK [, BYTE],      | set value of a CLI symbol
0605 1     VALUE: REF BLOCK [, BYTE],      | symbol string descriptor
0606 1     MODE: REF VECTOR [, LONG]       | desired symbol value descriptor
0607 1 ) =                                | desired symbol table (opt.)
0608 1
0609 1 ++
0610 1 FUNCTIONAL DESCRIPTION:
0611 1
0612 1     This routine defines a CLI symbol giving it the value specified by
0613 1     call parameter value. Before the actual definition, however, the
0614 1     specified symbol name is upcased so that it will better match symbol
0615 1     names formed by the CLI. An attempt to define a CLI symbol which
0616 1     already exists results in the value of that symbol being updated to
0617 1     the new value. The optional mode argument specifies a code
0618 1     (LIB$K_CLI_LOCAL_SYM or LIB$K_CLI_GLOBAL_SYM) which names the symbol
0619 1     table on which the define/update operation will be performed. If the
0620 1     mode argument is omitted, the the local symbol table is used.
0621 1
0622 1     The optional mode value can be one of:
0623 1         LIB$K_CLI_LOCAL_SYM = 1 ==> Local symbol table name
0624 1         LIB$K_CLI_GLOBAL_SYM = 2 ==> Global symbol table name
0625 1     These symbols are defined in $LIBCLIDEF.
0626 1
0627 1 CALLING SEQUENCE:
0628 1
0629 1     ret_status.wlc.v = LIB$SET_SYMBOL (symbol.rt.dx, value.rt.dx
0630 1                                     [, mode.rl.r])
0631 1
0632 1 FORMAL PARAMETERS:
0633 1
0634 1     symbol.rt.dx - A string, passed by descriptor, which contains the
0635 1                   symbol to be defined or modified. An upcased copy of
0636 1                   this string will actually be used for the define
0637 1                   or update operation.
0638 1
0639 1     value.rt.dx - A string, passed by descriptor, containing the value
0640 1                   to be given to the symbol.
0641 1
0642 1     mode.rl.r - An optional longword which, if present, contains a
0643 1                 value indicating into which table the symbol is to be
0644 1                 placed. If this parameter is omitted, the local
0645 1                 symbol table is used.
0646 1                 Possible values are:
0647 1                     LIB$K_CLI_LOCAL_SYM ==> Local symbol table name
0648 1                     LIB$K_CLI_GLOBAL_SYM ==> Global symbol table name
0649 1
0650 1 IMPLICIT INPUTS:
0651 1
0652 1     NONE
0653 1
0654 1 IMPLICIT OUTPUTS:
0655 1
0656 1     NONE
0657 1
0658 1 COMPLETION STATUS: (or ROUTINE VALUE:)

```



```
383 0659 1 1
384 0660 1 1
385 0661 1 1
386 0662 1 1
387 0663 1 1
388 0664 1 1
389 0665 1 1
390 0666 1 1
391 0667 1 1
392 0668 1 1
393 0669 1 1
394 0670 1 1
395 0671 1 1
396 0672 1 1
397 0673 1 1
398 0674 1 1
399 0675 1 1
400 0676 1 1
401 0677 1 1
402 0678 1 1
403 0679 1 1
404 0680 1 1
405 0681 1 1
406 0682 1 1
407 0683 1 1
408 0684 2 2
409 0685 2 2
410 0686 2 2
411 0687 2 2
412 0688 2 2
413 0689 2 2
414 0690 2 2
415 0691 2 2
416 0692 2 2
417 0693 2 2
418 0694 2 2
419 0695 2 2
420 0696 2 2
421 0697 2 2
422 0698 2 2
423 0699 2 2
424 0700 2 2
425 0701 2 2
426 0702 2 2
427 0703 2 2
428 0704 2 2
429 0705 2 2
430 0706 2 2
431 0707 2 2
432 0708 2 2
433 0709 2 2
434 0710 2 2
435 0711 2 2
436 0712 2 2
437 0713 2 2
438 0714 2 2
439 0715 2 2

SS$ NORMAL Normal successful completion
LIB$ INVARG Invalid argument
LIB$ INVSYMNAM Invalid symbol name
LIB$ INSCLIMEM Insufficient CLI memory
LIB$ AMBSYMDEF Ambiguous CLI symbol defined
LIB$ NOCLI No CLI present to perform function
LIB$ UNECLIERR Unexpected CLI Error

SIDE EFFECTS:

A CLI symbol is created or updated.

WARNING:
Although this procedure performs some checks on the validity of
symbol names passed to it, some symbol name considered valid by
this procedure will be considered invalid by DCL. Callers of this
procedure are responsible for insuring that symbol names passed to
this procedure contain only alphanumeric characters. If this
procedure is used to create symbols whose names contain invalid
characters, deleting those symbols can be accomplished by logging
out and logging back in.

--
BEGIN
BUILTIN
NULLPARAMETER;

LOCAL
CLI_REQ_BLOCK : BLOCK [CLISC_SRVDESC, BYTE], ! A CLI request block
DYN_STRING : BLOCK [8, BYTE], ! Descriptor for dynamic string
RETURN_STATUS : BLOCK [4, BYTE]; ! A return status value

+ Initialize CLI Command request block.
-
CH$FILL (0, CLISC_SRVDESC, CLI_REQ_BLOCK);
CLI_REQ_BLOCK [CLISB_RQTYPE] = CLISK_CLISERV;
BEGIN
LOCAL
STRING : REF BLOCK [8, BYTE];
STRING = .VALUE;
IF .STRING [DSC$W_LENGTH] GTRU 255 THEN RETURN LIB$ INVARG;
END;
MOVEDESC ( .VALUE, CLI_REQ_BLOCK [CLISQ_VALDESC] );
RETURN_STATUS = LIB$BUILT_SYMBOL_NAME ( .SYMBOL, DYN_STRING,
CLI_REQ_BLOCK [CLISQ_NAMDESC] );
IF NOT .RETURN_STATUS THEN RETURN .RETURN_STATUS;

+ Setup service code indicating which symbol table.
-
CLI_REQ_BLOCK [CLISW_SERVCOD] =
BEGIN
```

```

440      0716      3      IF NULLPARAMETER (3)
441      0717      THEN
442      0718      CLISK_DEFLOCAL
443      0719      ELSE
444      0720      CASE .MODE [0] FROM LIB$K_CLI_LOCAL_SYM
445      0721      TO LIB$K_CLI_GLOBAL_SYM OF
446      0722      SET
447      0723      [LIB$K_CLI_LOCAL_SYM] : CLISK_DEFLOCAL;
448      0724      [LIB$K_CLI_GLOBAL_SYM] : CLISK_DEFGLOBAL;
449      0725      [OUTRANGE] : RETURN (LIB$_INVARG);
450      0726      TES
451      0727      END;
452      0728
453      0729      +
454      0730      Set CLI symbol.
455      0731      -
456      0732      RETURN_STATUS = SYS$CLI (CLI_REQ_BLOCK);
457      0733
458      0734      +
459      0735      Free dynamic string used for upcased symbol name.
460      0736      -
461      0737      LIB$SFREE1_DD( DYN_STRING );
462      0738
463      0739      +
464      0740      Adjust error return status, if any.
465      0741      -
466      0742      IF NOT .RETURN_STATUS AND (.RETURN_STATUS [ST$V_FAC_NO] EQL CLIS_FACILITY)
467      0743      THEN
468      0744      RETURN_STATUS =
469      0745      BEGIN
470      0746      SELECTONE .RETURN_STATUS OF
471      0747      SET
472      0748      [CLIS_SYMOVF] : LIB$-INSCLIMEM;
473      0749      [CLIS_ABSYMD] : LIB$-AMBSYMDEF;
474      0750      [CLIS_INVREQTYP] : LIB$-NOCLI;
475      0751      [OTHERWISE] : LIB$-UNECLIERR;
476      0752      TES
477      0753      END;
478      0754
479      0755      RETURN (.RETURN_STATUS);
480      0756
481      0757      END;

```

! End of routine LIB\$SET_SYMBOL

0054	BF	00	5E	A4	AE	003C 00000	.ENTRY	LIB\$SET_SYMBOL, Save R2,R3,R4,R5	0603
			6E		00	9E 00002	MOVAB	-92(SP), SP	
				08	00	2C 00006	MOVCS	#0, (SP), #0, #84, CLI_REQ_BLOCK	0698
		08	AE		05	90 0000F	MOVB	#5, CLI_REQ_BLOCK	0699
			50	08	AC	D0 00013	MOVL	VALUE, STRING	0703
	00FF		8F		60	B1 00017	CMPW	(STRING), #255	0704
					3D	1A 0001C	BGTRU	3\$	
		50		08	AC	D0 0001E	MOVL	VALUE, R0	0706
					00	16 00022	JSB	LIB\$ANALYZE_SDESC_R2	

14	AE	51	B0	00028	MOVW	R1, CLI_REQ_BLOCK+12	
18	AE	52	D0	0002C	MOVL	R2, CLI_REQ_BLOCK+16	
	01	50	E8	00030	BLBS	RET_STATUS, -1\$	
			04	00033	RET		
		0C	AE	9F	00034	1\$: PUSHAB	CLI_REQ_BLOCK+4
		04	AE	9F	00037	PUSHAB	DYN_STRING
		04	AC	DD	0003A	PUSHL	SYMBOL
0000V	CF	03	FB	0003D	CALLS	#3, LIB\$BUILD_SYMBOL_NAME	
	52	50	D0	00042	MOVL	R0, RETURN_STATUS	
	7B	52	E9	00045	BLBC	RETURN_STATUS, 9\$	0709
	03	6C	91	00048	CMPB	(AP), #3	0716
		16	1F	0004B	BLSSU	4\$	
		0C	AC	D5	0004D	TSTL	12(AP)
		11	13	00050	BEQL	4\$	
01	01	0C	BC	CF	00052	CASEL	@MODE, #1, #1
	0011	000C	CF	00057	.WORD	4\$-2\$, -	0720
						5\$-2\$	
	50	00000000G	8F	D0	0005B	3\$: MOVL	#LIB\$_INVARG, R0
				04	00062	RET	
	50		02	D0	00063	4\$: MOVL	#2, R0
			03	11	00066	BRB	6\$
	50		03	D0	00068	5\$: MOVL	#3, R0
09	AE		50	B0	0006B	6\$: MOVW	R0, CLI_REQ_BLOCK+1
		08	AE	9F	0006F	PUSHAB	CLI_REQ_BLOCK
00000000G	00		01	FB	00072	CALLS	#1, SYS\$CLI
	52		50	D0	00079	MOVL	R0, RETURN_STATUS
			5E	DD	0007C	PUSHL	SP
00000000G	00		01	FB	0007E	CALLS	#1, LIB\$FREE1_DD
	44		52	E8	00085	BLBS	RETURN_STATUS, -11\$
03	52		10	ED	00088	CMPZV	#16, #T2, RETURN_STATUS, #3
			3D	12	0008D	BNEQ	11\$
00038138	8F		52	D1	0008F	CMPL	RETURN_STATUS, #229688
			09	12	00096	BNEQ	7\$
	52	00000000G	8F	D0	00098	MOVL	#LIB\$_INSCLIMEM, RETURN_STATUS
			2B	11	0009F	BRB	11\$
000381A0	8F		52	D1	000A1	7\$: CMPL	RETURN_STATUS, #229792
			09	12	000A8	BNEQ	8\$
	52	00000000G	8F	D0	000AA	MOVL	#LIB\$_AMBSYMDEF, RETURN_STATUS
			19	11	000B1	BRB	11\$
00038822	8F		52	D1	000B3	8\$: CMPL	RETURN_STATUS, #231458
			09	12	000BA	BNEQ	10\$
	52	00000000G	8F	D0	000BC	MOVL	#LIB\$_NOCLI, RETURN_STATUS
			07	11	000C3	BRB	11\$
	52	00000000G	8F	D0	000C5	10\$: MOVL	#LIB\$_UNECLIERR, RETURN_STATUS
	50		52	D0	000CC	11\$: MOVL	RETURN_STATUS, R0
			04	000CF	RET		0751
							0755
							0757

: Routine Size: 208 bytes. Routine Base: _LIB\$CODE + 00D5

```

483 0758 1 XSBTTL 'LIB$DELETE_SYMBOL - delete a CLI symbol'
484 0759 1 GLOBAL ROUTINE LIB$DELETE_SYMBOL (
485 0760 1     SYMBOL: REF BLOCK [, BYTE],
486 0761 1     MODE: REF VECTOR [, LONG]
487 0762 1 ) =
488 0763 1
489 0764 1 ++
490 0765 1 FUNCTIONAL DESCRIPTION:
491 0766 1
492 0767 1     This routine deletes a CLI symbol. Before the actual deletion,
493 0768 1     however, the specified symbol name is upcased so that it will better
494 0769 1     match symbol names formed by the CLI. The optional mode argument
495 0770 1     specifies a code (LIB$K_CLI_LOCAL_SYM or LIB$K_CLI_GLOBAL_SYM) which
496 0771 1     names the symbol table on which the delete operation will be performed.
497 0772 1     If the mode argument is omitted, the the local symbol table is used.
498 0773 1
499 0774 1     The optional mode value can be one of:
500 0775 1         LIB$K_CLI_LOCAL_SYM = 1 ==> Local symbol table name
501 0776 1         LIB$K_CLI_GLOBAL_SYM = 2 ==> Global symbol table name
502 0777 1     These symbols are defined in $LIBCLIDEF.
503 0778 1
504 0779 1 CALLING SEQUENCE:
505 0780 1
506 0781 1     ret_status.wlc.v = LIB$DELETE_SYMBOL (symbol.rt.dx [, mode.rl.r])
507 0782 1
508 0783 1 FORMAL PARAMETERS:
509 0784 1
510 0785 1     symbol.rt.dx - A string, passed by descriptor, which contains the
511 0786 1                   symbol to be defined or modified. An upcased copy of
512 0787 1                   this string will actually be used for the deletion.
513 0788 1
514 0789 1     mode.rl.r - An optional longword which, if present, contains a
515 0790 1                  value indicating from which table the symbol is to be
516 0791 1                  deleted. If this parameter is omitted, the local
517 0792 1                  symbol table is used.
518 0793 1                  Possible values are:
519 0794 1                      LIB$K_CLI_LOCAL_SYM ==> Local symbol table name
520 0795 1                      LIB$K_CLI_GLOBAL_SYM ==> Global symbol table name
521 0796 1
522 0797 1 IMPLICIT INPUTS:
523 0798 1
524 0799 1     NONE
525 0800 1
526 0801 1 IMPLICIT OUTPUTS:
527 0802 1
528 0803 1     NONE
529 0804 1
530 0805 1 COMPLETION STATUS: (or ROUTINE VALUE:)
531 0806 1
532 0807 1     SSS NORMAL      Normal successful completion
533 0808 1     LIB$INVARG      Invalid argument
534 0809 1     LIB$INVSYMNAM   Invalid symbol name
535 0810 1     LIB$NOSUCHSYM   No such symbol found
536 0811 1     LIB$NOCLI       No CLI present to perform function
537 0812 1     LIB$UNECLIERR   Unexpected CLI Error
538 0813 1
539 0814 1 SIDE EFFECTS:

```



```
0815 1 |
0816 1 | A CLI symbol is deleted.
0817 1 |
0818 1 | WARNING:
0819 1 | Although this procedure performs some checks on the validity of
0820 1 | symbol names passed to it, some symbol name considered valid by
0821 1 | this procedure will be considered invalid by DCL. Callers of this
0822 1 | procedure are responsible for insuring that symbol names passed to
0823 1 | this procedure contain only alphanumeric characters.
0824 1 |
0825 1 |
0826 1 |
0827 2 | BEGIN
0828 2 |
0829 2 | BUILTIN
0830 2 |     NULLPARAMETER;
0831 2 |
0832 2 | LOCAL
0833 2 |     CLI_REQ_BLOCK : BLOCK [CLISC_SRVDESC, BYTE], ! A CLI request block
0834 2 |     DYN_STRING : BLOCK [8, BYTE], ! Descriptor for dynamic string
0835 2 |     RETURN_STATUS : BLOCK [4, BYTE]; ! A return status value
0836 2 |
0837 2 |
0838 2 |
0839 2 | + Initialize CLI Command request block.
0840 2 | -
0841 2 |     CH$FILL (0, CLISC_SRVDESC, CLI_REQ_BLOCK);
0842 2 |     RETURN_STATUS = LIB$BUILD_SYMBOL_NAME( .SYMBOL, DYN_STRING,
0843 2 |         CLI_REQ_BLOCK[CLISC_NAMDESC] );
0844 2 |     IF NOT .RETURN_STATUS THEN RETURN .RETURN_STATUS;
0845 2 |     CLI_REQ_BLOCK[CLISC_RQTYPE] = CLISK_CLISERV;
0846 2 |
0847 2 | +
0848 2 | + Setup service code indicating which symbol table.
0849 2 | -
0850 2 |     CLI_REQ_BLOCK[CLISW_SERVCOD] =
0851 2 |     BEGIN
0852 2 |         IF NULLPARAMETER (2)
0853 2 |         THEN
0854 2 |             CLISK_DELELCL
0855 2 |         ELSE
0856 2 |             CASE .MODE [0] FROM LIB$K_CLI_LOCAL_SYM
0857 2 |                 TO LIB$K_CLI_GLOBA_SYM OF
0858 2 |                 SET
0859 2 |                 [LIB$K_CLI_LOCAL_SYM] : CLISK_DELELCL;
0860 2 |                 [LIB$K_CLI_GLOBA_SYM] : CLISK_DELEGBL;
0861 2 |                 [OUTRANGE] : RETURN (LIB$INVARG);
0862 2 |             TES
0863 2 |         END;
0864 2 |
0865 2 | +
0866 2 | + Delete CLI symbol.
0867 2 | -
0868 2 |     RETURN_STATUS = SYS$CLI (CLI_REQ_BLOCK);
0869 2 |
0870 2 | +
0871 2 | + Free dynamic string used for upcased symbol name.
```

```
0872 2 !-
0873 2 LIB$FREE1_DD( DYN_STRING );
0874 2
0875 2 !+
0876 2 Adjust error return status, if any.
0877 2
0878 2 IF NOT .RETURN_STATUS AND (.RETURN_STATUS [STS$V_FAC_NO] EQL CLI$FACILITY)
0879 2 THEN
0880 2     RETURN_STATUS =
0881 2     BEGIN
0882 2         SELECT ONE .RETURN_STATUS OF
0883 2         SET
0884 2         [CLI$_UNDSYM] : LIB$NOSUCHSYM;
0885 2         [CLI$_INVREQ_TYP] : LIB$NOCLI;
0886 2         [OTHERWISE] : LIB$UNECLERR;
0887 2         YES
0888 2         END;
0889 2
0890 2 RETURN (.RETURN_STATUS);
0891 2
0892 1 END;
```

! End of routine LIB\$DELETE_SYMBOL

0054	BF	00	5E	A4	AE	9E	00002	.ENTRY	LIB\$DELETE_SYMBOL, Save R2,R3,R4,R5	0759	
			6E	08	00	2C	00006	MOVAB	-92(SP), SP		
				0C	AE	9F	0000D	MOVCS	#0, (SP), #0, #84, CLI_REQ_BLOCK	0841	
				04	AE	9F	00012	PUSHAB	CLI_REQ_BLOCK+4	0843	
				04	AC	DD	00015	PUSHAB	DYN_STRING	0842	
				03	FB	00018	PUSHL	SYMBOL		0843	
	0000V	CF		50	D0	0001D	CALLS	#3, LIB\$BUILD_SYMBOL_NAME			
		52		52	E9	00020	MOVL	R0, RETURN_STATUS			
		76		05	90	00023	BLBC	RETURN_STATUS, 7\$	0844		
	08	AE		6C	91	00027	MOVB	#5, CLI_REQ_BLOCK	0845		
		02		16	1F	0002A	CMPB	(AP), #2	0852		
				08	AC	D5	0002C	BLSSU	2\$		
				11	13	0002F	TSTL	8(AP)			
	01	01		08	BC	CF	00031	BEQL	2\$		
		0011		000C		00036	CASEL	@MODE, #1, #1	0856		
							.WORD	2\$-1\$,-			
								3\$-1\$			
		50	00000000G	8F	D0	0003A	MOVL	#LIB\$INVARG, R0	0861		
				04	04	00041	RET				
		50		0B	D0	00042	MOVL	#11, R0	0856		
				03	11	00045	BRE	4\$			
		50		0C	D0	00047	MOVL	#12, R0			
	09	AE		50	B0	0004A	MOVW	R0, CLI_REQ_BLOCK+1	0851		
				08	AE	9F	0004E	PUSHAB	CLI_REQ_BLOCK	0868	
		00000000G	00	01	FB	00051	CALLS	#1, SYS\$CLI			
			52	50	D0	00058	MOVL	R0, RETURN_STATUS			
				5E	DD	0005B	PUSHL	SP	0873		
		00000000G	00	01	FB	0005D	CALLS	#1, LIB\$FREE1_DD			
			32	52	E8	00064	BLBS	RETURN_STATUS, 7\$	0878		
03		52	0C	10	ED	00067	CMPZV	#16, #T2, RETURN_STATUS, #3			

00038140	8F	28	12	0006C	BNEQ	7\$		
		52	D1	0006E	CMPL	RETURN_STATUS, #229696	...	0884
		09	12	00075	BNEQ	5\$		
	52 00000000G	8F	D0	00077	MOVL	#LIB\$_NOSUCHSYM, RETURN_STATUS		
		19	11	0007E	BRB	7\$		
00038822	8F	52	D1	00080	CMPL	RETURN_STATUS, #231458	...	0885
		09	12	00087	BNEQ	6\$		
	52 00000000G	8F	D0	00089	MOVL	#LIB\$_NOCLI, RETURN_STATUS		
		07	11	00090	BRB	7\$		
	52 00000000G	8F	D0	00092	MOVL	#LIB\$_UNECLIERR, RETURN_STATUS	...	0886
	50	52	D0	00099	MOVL	RETURN_STATUS, R0	...	0890
		04		0009C	RET		...	0892

; Routine Size: 157 bytes, Routine Base: _LIB\$CODE + 01A5


```
0893 1 XSBTTL 'LIB$SET LOGICAL - set supervisor mode logical name'
0894 1 GLOBAL ROUTINE [LIB$SET LOGICAL ( set supervisor mode logical name
0895 1 LOGNAME : REF $BBLOCK, logical name string descriptor
0896 1 VALUE : REF $BBLOCK, equivalence name string descriptor
0897 1 TABNAM : REF $BBLOCK, logical name table
0898 1 ATTR : REF $BBLOCK[4], attributes
0899 1 ITMLST : REF $BBLOCK Address of itemlist
0900 1 ) =
0901 1
0902 1 ++
0903 1 FUNCTIONAL DESCRIPTION:
0904 1
0905 1 This routine provides a method by which a non-privileged program can
0906 1 create/modify supervisor mode logical names. The logical name given
0907 1 by logname is created or modified to have the equivalence name
0908 1 specified by value.
0909 1 The user may specify attributes for the logical name and may
0910 1 specify which logical name table is to be used.
0911 1 Instead of specifying a single equivalence string, via VALUE,
0912 1 the caller may specify an item list to give multiple equivalence strings.
0913 1
0914 1 The $TRNLNM system service can be used to obtain the value of
0915 1 supervisor mode logical names. The routine is equivalent to the ASSIGN
0916 1 or DEFINE DCL command.
0917 1
0918 1 CALLING SEQUENCE:
0919 1
0920 1 ret_status.wlc.v = LIB$SET_LOGICAL (
0921 1 logname.rt.dx
0922 1 [, value.rt.dx]
0923 1 [, tabnam.rt.dx]
0924 1 [, attr.rlu.r]
0925 1 [, itmlst.ra.r])
0926 1
0927 1 FORMAL PARAMETERS:
0928 1
0929 1 logname.rt.dx - A string, passed by descriptor, which contains the
0930 1 supervisor mode logical name to be created or modified.
0931 1
0932 1 value.rt.dx - A string, passed by descriptor, specifying the value
0933 1 to be given to the supervisor mode logical name.
0934 1 If omitted, an itemlist must be present to specify the
0935 1 value(s) of the logical name.
0936 1
0937 1 tabnam.rt.dx - Name of the table in which to create the logical name.
0938 1 If omitted, the LNMSPROCESS table is used.
0939 1
0940 1 attr.rlu.r - Logical name attributes. See the description of
0941 1 the system service $CRELNM for more details.
0942 1 Currently available attributes are
0943 1 LNMSM CONFINED and LNMSM NO ALIAS.
0944 1 If omitted, no special attributes are established.
0945 1 However, if no table or itemlist is specified,
0946 1 then the attribute LNMSM_CRELOG is defaulted in.
0947 1
0948 1 If no itemlist is specified, then the translation
0949 1 attributes LNMSM_CONCEALED and LNMSM_TERMINAL may
```

```
676 0950 1 also be specified here. They apply to the
677 0951 1 equivalence string string specified by the VALUE
678 0952 1 parameter.
679 0953 1
680 0954 1 If an itemlist is specified, then the itemlist will
681 0955 1 contain the translation attributes for each
682 0956 1 equivalence string in the itemlist.
683 0957 1
684 0958 1 itmlst.ra.r The address of an item list describing the
685 0959 1 equivalence name(s) for this logical name.
686 0960 1 See the description of the $CRELNM system service
687 0961 1 for the format and meaning of the item list.
688 0962 1
689 0963 1 If omitted, the logical name will have just one
690 0964 1 value, as specified by the VALUE parameter.
691 0965 1
692 0966 1 Either VALUE or ITMLST must be specified.
693 0967 1 If neither are specified, the LIBS_INVARG
694 0968 1 error is produced.
695 0969 1
696 0970 1 If both are specified, then the VALUE parameter
697 0971 1 is ignored. In this case, logical name
698 0972 1 attributes are permitted, but translation
699 0973 1 attributes specified in the ATTR parameter
700 0974 1 will be ignored (since these will appear in the
701 0975 1 itemlist).
702 0976 1
703 0977 1 The ITMLST parameter is only needed in the case where
704 0978 1 you wish to create multiple equivalence strings
705 0979 1 for a single logical name (i.e. a search list).
706 0980 1
707 0981 1 IMPLICIT INPUTS:
708 0982 1
709 0983 1 It is assumed that the logical name table specified already exists.
710 0984 1
711 0985 1 IMPLICIT OUTPUTS:
712 0986 1
713 0987 1 NONE
714 0988 1
715 0989 1 COMPLETION STATUS: (or ROUTINE VALUE:)
716 0990 1
717 0991 1 SSS_NORMAL Normal successful completion
718 0992 1 SSS_SUPERSEDE Previous logical name replaced (success)
719 0993 1 SSS_ACCVIO logname or value cannot be read
720 0994 1 SSS_IVLOGNAM logname or value contains more than 255 characters
721 0995 1 SSS_BADPARAM One or more arguments had a bad value.
722 0996 1 For example, an unknown bit was specified in
723 0997 1 the attributes parameter.
724 0998 1 Note that if an item list is specified,
725 0999 1 then translation attributes (if any) must
726 1000 1 appear in the item list and may not appear
727 1001 1 in the ATTR parameter. Only logical name
728 1002 1 attributes may appear in the ATTR parameter
729 1003 1 in that case.
730 1004 1 SSS_NOLOGNAM The specified logical name table does not exist.
731 1005 1 SSS_NOPRIV The caller lacks the necessary privileges
732 1006 1 to create the logical name.
```

```

733 1007 1  SSS_TOOMANYLNAM Logical name translation of the table name exceeded
734 1008 1  the allowable depth (10 translations).
735 1009 1  SSS_DUPLNAM Attempt to create a logical name that conflicts
736 1010 1  with a logical name created at a higher access mode
737 1011 1  and which had the LNMSM_NO_ALIAS attribute.
738 1012 1  SSS_EXLNMQOTA Exceeded logical name space quota.
739 1013 1  SSS_INSMEM Insufficient dynamic memory.
740 1014 1  LIB$INVARG Invalid argument. Neither a VALUE nor an ITMLST
741 1015 1  was specified.
742 1016 1  Or LOGNAME was omitted.
743 1017 1  LIB$NOCLI No CLI present to perform function
744 1018 1  LIB$UNECLIERR Unexpected CLI Error
745 1019 1  LIB$WRONUMARG Wrong number of arguments. Either fewer than two
746 1020 1  arguments or more than 5 arguments were specified.
747 1021 1
748 1022 1  SIDE EFFECTS:
749 1023 1
750 1024 1  A supervisor mode logical name is created or modified.
751 1025 1
752 1026 1  --
753 1027 1
754 1028 2  BEGIN
755 1029 2
756 1030 2  BUILTIN
757 1031 2
758 1032 2  ACTUALCOUNT,
759 1033 2  NULLPARAMETER;
760 1034 2
761 1035 2  LOCAL
762 1036 2
763 1037 2  CLI_REQ_BLOCK : BLOCK [CLISC_SRVDESC, BYTE], ! A CLI request block
764 1038 2  RETURN_STATUS : BLOCK [4, BYTE],
765 1039 2  TRANSLATION_ATTRIBUTES,
766 1040 2  LOGICAL_NAME_ATTRIBUTES,
767 1041 2  SPECIAL_ITEM_LIST : VECTOR [7]; ! Special item list
768 1042 2
769 1043 2  +
770 1044 2  Check for proper number of arguments.
771 1045 2  -
772 1046 2
773 1047 2  IF ACTUALCOUNT() LEQU 1
774 1048 2  OR ACTUALCOUNT() GTRU 5
775 1049 2  THEN
776 1050 2  RETURN LIB$WRONUMARG;
777 1051 2
778 1052 2  +
779 1053 2  Initialize CLI Command request block accounting for the differences
780 1054 2  between creating/modifying a logical name and deleting one.
781 1055 2  -
782 1056 2
783 1057 2  CH$FILL (0, CLISC_SRVDESC, CLI_REQ_BLOCK);
784 1058 2  CLI_REQ_BLOCK [CLISB_RQTYPE] = CLISK_CLISERV;
785 1059 2  CLI_REQ_BLOCK [CLISW_SERVCOD] = CLISK_CREALOG;
786 1060 2
787 1061 2  +
788 1062 2  Move in the logical name, the equivalence string, and the logical
789 1063 2  name table name.

```



```
790 1064 2 | Give an error if the logical name is omitted.
791 1065 2 | Give an error if neither VALUE nor ITMLST is supplied.
792 1066 2 |
793 1067 2 |
794 1068 2 | IF .LOGNAME EQL 0
795 1069 2 | THEN
796 1070 2 |     RETURN LIB$_INVARG;
797 1071 2 |
798 1072 2 | MOVEDESC ( .LOGNAME, CLI_REQ_BLOCK [CLISQ_NAMDESC] );
799 1073 2 |
800 1074 2 | IF NULLPARAMETER(VALUE) AND NULLPARAMETER(ITMLST)
801 1075 2 | THEN
802 1076 2 |     RETURN LIB$_INVARG;
803 1077 2 |
804 1078 2 | IF NOT NULLPARAMETER(VALUE)
805 1079 2 | THEN
806 1080 2 |     MOVEDESC ( .VALUE, CLI_REQ_BLOCK [CLISQ_VALDESC] );
807 1081 2 |
808 1082 2 | +
809 1083 2 | | NOTE:
810 1084 2 | | If both VALDESC and ITMLST are specified, DCL will ignore VALUE.
811 1085 2 | |
812 1086 2 | |
813 1087 2 | |
814 1088 2 | | Move in the table name if present.
815 1089 2 | | If not present, DCL defaults to LNMSPROCESS.
816 1090 2 | |
817 1091 2 | |
818 1092 2 | IF NOT NULLPARAMETER(TABNAM)
819 1093 2 | THEN
820 1094 2 |     MOVEDESC ( .TABNAM, CLI_REQ_BLOCK [CLISQ_TABDESC] );
821 1095 2 |
822 1096 2 | +
823 1097 2 | | Move in the address of the item list (if present).
824 1098 2 | | If not present, DCL will create an itemlist from CLISQ_VALDESC.
825 1099 2 | |
826 1100 2 | |
827 1101 2 | IF NOT NULLPARAMETER(ITMLST)
828 1102 2 | THEN
829 1103 2 |     CLI_REQ_BLOCK [CLISL_ITMLST] = .ITMLST;
830 1104 2 |
831 1105 2 | +
832 1106 2 | | Move in the address of the attributes (if present).
833 1107 2 | |
834 1108 2 | |
835 1109 2 | IF NOT NULLPARAMETER(ATTR)
836 1110 2 | THEN
837 1111 2 |     CLI_REQ_BLOCK [CLISL_ATTR] = .ATTR;
838 1112 2 |
839 1113 2 | +
840 1114 2 | | NOTE:
841 1115 2 | | If CLISQ_TABDESC, CLISL_ITMLST, and CLISL_ATTR are all 0,
842 1116 2 | | then DCL defaults in the attribute LNMSM_CRELOG, i.e.
843 1117 2 | | the call back looks like a V3 call back.
844 1118 2 | |
845 1119 2 | |
846 1120 2 | +
```

```

847 1121 2  If the caller has specified translation attributes in ATTR
848 1122 2  (as opposed to just logical name attributes), this is normally
849 1123 2  an error. That is because translation attributes should go in
850 1124 2  the itemlist. However, as a convenience to RTL users who find
851 1125 2  it hard to build itemlists, we allow translation attributes to
852 1126 2  occur if no itemlist is specified. In that case, we have to go
853 1127 2  through the trouble of building an itemlist for him.
854 1128 2  -
855 1129 2
856 1130 2  IF NULLPARAMETER(ITMLST) AND NOT NULLPARAMETER(ATTR)
857 1131 2  THEN
858 1132 2  BEGIN ! Check for translation attributes
859 1133 2  BIND ATTR_VECTOR = .ATTR : VECTOR[4,BYTE],
860 1134 2  ATTR_VALUE = .ATTR : LONG;
861 1135 2
862 1136 2  +
863 1137 2  The first byte of attributes are logical name attributes.
864 1138 2  The second byte of attributes are translation attributes.
865 1139 2  The other 2 bytes are reserved for VMS use.
866 1140 2  -
867 1141 2
868 1142 2  IF .ATTR_VECTOR[1] NEQ 0
869 1143 2  THEN
870 1144 4  BEGIN ! Build special itemlist
871 1145 4  BIND EQUIV_STRING_DESC = CLI_REQ_BLOCK[CLISQ_VALDESC] : $BLOCK;
872 1146 4
873 1147 4  +
874 1148 4  Create translation attributes from the specified attributes.
875 1149 4  -
876 1150 4
877 1151 4  TRANSLATION_ATTRIBUTES=.ATTR_VALUE AND %X'FF00';
878 1152 4
879 1153 4  +
880 1154 4  Create logical name attributes from the specified attributes.
881 1155 4  -
882 1156 4
883 1157 4  LOGICAL_NAME_ATTRIBUTES=.ATTR_VALUE AND %X'FF';
884 1158 4
885 1159 4  SPECIAL_ITEM_LIST[0]=LNMS_ATTRIBUTES*16+4;
886 1160 4  SPECIAL_ITEM_LIST[1]=TRANSLATION_ATTRIBUTES;
887 1161 4  SPECIAL_ITEM_LIST[2]=0;
888 1162 4  SPECIAL_ITEM_LIST[3]=LNMS_STRING*16+.EQUIV_STRING_DESC[DSC$W_LENGTH];
889 1163 4  SPECIAL_ITEM_LIST[4]=.EQUIV_STRING_DESC[DSC$A_POINTER];
890 1164 4  SPECIAL_ITEM_LIST[5]=0;
891 1165 4  SPECIAL_ITEM_LIST[6]=0; ! no more entries
892 1166 4
893 1167 4  +
894 1168 4  Store specially revised attributes and item list in
895 1169 4  the request block.
896 1170 4  -
897 1171 4
898 1172 4  CLI_REQ_BLOCK[CLISL_ATTR]=LOGICAL_NAME_ATTRIBUTES;
899 1173 4  CLI_REQ_BLOCK[CLISL_ITMLST]=SPECIAL_ITEM_LIST
900 1174 4
901 1175 4  END ! Build special itemlist
902 1176 4
903 1177 2  END: ! Check for translation attributes
```

```

904      1178      2
905      1179      2
906      1180      2
907      1181      2
908      1182      2
909      1183      2
910      1184      2
911      1185      2
912      1186      2
913      1187      2
914      1188      2
915      1189      2
916      1190      2
917      1191      2
918      1192      2
919      1193      2
920      1194      2
921      1195      2
922      1196      2
923      1197      2
924      1198      2
925      1199      2
926      1200      2

      + Create or modify the supervisor mode logical name.
      RETURN_STATUS = SY$CLI (CLI_REQ_BLOCK);

      + Adjust error return status, if any.
      IF NOT .RETURN_STATUS AND (.RETURN_STATUS [ST$V_FAC_NO] EQL CLI$FACILITY)
      THEN
          RETURN_STATUS =
          BEGIN
              SELECTONE .RETURN_STATUS OF
              SET
              [CLI$INVREQTYP] : LIB$NOCLI;
              [OTHERWISE] : LIB$UNECLIERR;
              TES
          END;

      RETURN .RETURN_STATUS

      END;

```

```

! End of LIB$SET_LOGICAL

```

				007C 00000	.ENTRY LIB\$SET_LOGICAL, Save R2,R3,R4,R5,R6	0894
	56	00000000G	00	9E 00002	MOVAB LIB\$ANALYZE_SDESC_R2, R6	
	5E	88	AE	9E 00009	MOVAB -120(SP), SP	
	01		6C	91 0000D	CMPB (AP), #1	1047
			05	1B 00010	BLEQU 1\$	
	05		6C	91 00012	CMPB (AP), #5	1048
			08	1B 00015	BLEQU 2\$	
	50	00000000G	8F	D0 00017 1\$:	MOVL #LIB\$WRONUMARG, R0	1050
				04 0001E	RET	
0054	8F	00	6E	2C 0001F 2\$:	MOVCS #0, (SP), #0, #84, CLI_REQ_BLOCK	1057
		24	AE	05 00026	MOVAB #5, CLI_REQ_BLOCK	1058
	24	AE	06	B0 00028	MOVW #6, CLI_REQ_BLOCK+1	1059
		04	AC	D5 0002C	TSTL LOGNAME	1068
			25	13 00033	BEQL 4\$	
	50	04	AC	D0 00035	MOVL LOGNAME, R0	1072
			66	16 00039	JSB LIB\$ANALYZE_SDESC_R2	
28	AE		51	B0 0003B	MOVW R1, CLI_REQ_BLOCK+4	
2C	AE		52	D0 0003F	MOVL R2, CLI_REQ_BLOCK+8	
	4F		50	E9 00043	BLBC RET STATUS, 7\$	
	02		6C	91 00046	CMPB (APT, #2	1074
		08	05	1F 00049	BLSSU 3\$	
			AC	D5 0004B	TSTL 8(AP)	
			12	12 0004E	BNEQ 5\$	
	05		6C	91 00050 3\$:	CMPB (AP), #5	
		14	05	1F 00053	BLSSU 4\$	
			AC	D5 00055	TSTL 20(AP)	
			08	12 00058	BNEQ 5\$	
	50	00000000G	8F	D0 0005A 4\$:	MOVL #LIB\$INVARG, R0	1076

				04	00061		RET				
	02		6C	91	00062	5\$:	CMPB	(AP), #2		1078	
			16	1F	00063		BLSSU	6\$			
		08	AC	D5	00067		TSTL	8(AP)			
			11	13	0006A		BEQL	6\$			
	50		AC	D0	0006C		MOVL	VALUE, R0		1080	
			66	16	00070		JSB	LIB\$ANALYZE_SDESC R2			
30	AE		51	B0	00072		MOVW	R1, CLI_REQ_BLOCK+12			
34	AE		52	D0	00076		MOVL	R2, CLI_REQ_BLOCK+16			
	18		50	E9	0007A		BLBC	RET STATUS, -7\$			
	03		6C	91	0007D	6\$:	CMPB	(APT), #3		1092	
			17	1F	00080		BLSSU	8\$			
		0C	AC	D5	00082		TSTL	12(AP)			
			12	13	00085		BEQL	8\$			
	50		AC	D0	00087		MOVL	TABNAM, R0		1094	
			66	16	0008B		JSB	LIB\$ANALYZE_SDESC R2			
38	AE		51	B0	0008D		MOVW	R1, CLI_REQ_BLOCK+20			
3C	AE		52	D0	00091		MOVL	R2, CLI_REQ_BLOCK+24			
	01		50	E8	00095	7\$:	BLBS	RET STATUS, -8\$			
				04	00098		RET				
	05		6C	91	00099	8\$:	CMPB	(AP), #5		1101	
			0A	1F	0009C		BLSSU	9\$			
		14	AC	D5	0009E		TSTL	20(AP)			
			05	13	000A1		BEQL	9\$			
40	AE		AC	D0	000A3		MOVL	ITMLST, CLI_REQ_BLOCK+28		1103	
	04		6C	91	000A8	9\$:	CMPB	(AP), #4		1109	
			0A	1F	000AB		BLSSU	10\$			
		10	AC	D5	000AD		TSTL	16(AP)			
			05	13	000B0		BEQL	10\$			
44	AE		AC	D0	000B2		MOVL	ATTR, CLI_REQ_BLOCK+32		1111	
	05		6C	91	000B7	10\$:	CMPB	(AP), #5		1130	
			05	1F	000BA		BLSSU	11\$			
		14	AC	D5	000BC		TSTL	20(AP)			
			4B	12	000BF		BNEQ	12\$			
	04		6C	91	000C1	11\$:	CMPB	(AP), #4			
			46	1F	000C4		BLSSU	12\$			
		10	AC	D5	000C6		TSTL	16(AP)			
			41	13	000C9		BEQL	12\$			
	50		AC	D0	000CB		MOVL	ATTR, R0		1133	
		01	A0	95	000CF		TSTB	1(R0)		1142	
			38	13	000D2		BEQL	12\$			
6E	10	BC	FFFF00FF	8F	CB	000D4	BICL3	#-65281, @ATTR, TRANSLATION_ATTRIBUTES		1151	
	04	AE		BC	9A	000DD	MOVZBL	@ATTR, LOGICAL_NAME_ATTRIBUTES		1157	
	08	AE	00030004	8F	D0	000E2	MOVL	#196612, SPECIAL_ITEM_LIST		1159	
	0C	AE		6E	9E	000EA	MOVAB	TRANSLATION_ATTRIBUTES, SPECIAL_ITEM_LIST+4		1160	
				AE	D4	000EE	CLRL	SPECIAL_ITEM_LIST+8		1161	
	14	AE		AE	3C	000F1	MOVZWL	EQUIV_STRING_DESC, SPECIAL_ITEM_LIST+12		1162	
	16	AE		02	A0	000F6	ADDW2	#2, SPECIAL_ITEM_LIST+12			
	18	AE		AE	D0	000FA	MOVL	EQUIV_STRING_DESC+4, SPECIAL_ITEM_LIST+16		1163	
				1C	AE	7C	CLRL	SPECIAL_ITEM_LIST+20		1164	
	44	AE		AE	9E	00102	MOVAB	LOGICAL_NAME_ATTRIBUTES, CLI_REQ_BLOCK+32		1172	
	40	AE		AE	9E	00107	MOVAB	SPECIAL_ITEM_LIST, CLI_REQ_BLOCK+28		1173	
				AE	9F	0010C	PUSHAB	CLI_REQ_BLOCK		1182	
	00000000G	00		01	FB	0010F	CALLS	#1-SYS\$CLI			
		1F		50	E8	00116	BLBS	RETURN STATUS, 14\$		1187	
03	50	0C		10	ED	00119	CMPZV	#16, #T2, RETURN_STATUS, #3			
				1B	12	0011E	BNEQ	14\$			

LIBSCLI_CALLBACK LIBSCLI_CALLBACK - CLI Callback Interface Proce 16-Sep-1984 02:22:35 VAX-11 BLISS-32 V4.0-742
 V04-000 LIB\$SET_LOGICAL - set supervisor mode logical n 14-Sep-1984 13:27:47 [VMSLIB.SRC]LIBCLICAL.B32;1

Page 25
(6)

00038822	8F	50	D1	00120	CML	RETURN_STATUS, #231458	:	1193
		08	12	00127	BNEQ	13\$:	
	50	00000000G	8F	D0	00129	MOVL	#LIBS_NOCLI, RETURN_STATUS	:
			04	00130	RET		:	
	50	00000000G	8F	D0	00131	MOVL	#LIBS_UNECLIERR, RETURN_STATUS	:
			04	00138	RET		:	1194
							:	1200

; Routine Size: 313 bytes, Routine Base: _LIB\$CODE + 0242

```

928 1201 1 %SBTTL 'LIB$DELETE LOGICAL - delete supervisor mode logical name'
929 1202 1 GLOBAL ROUTINE LIB$DELETE_LOGICAL (      ! delete supervisor mode logical name
930 1203 1      LOGNAME : REF $BBLOCK,      ! logical name string descriptor
931 1204 1      TABNAM  : REF $BBLOCK      ! table name
932 1205 1      ) =
933 1206 1
934 1207 1 ++
935 1208 1 FUNCTIONAL DESCRIPTION:
936 1209 1
937 1210 1     This routine provides a method by which a non-privileged program can
938 1211 1     delete a supervisor mode logical name. The logical name given by
939 1212 1     logname is deleted from the specified or implied logical name table.
940 1213 1
941 1214 1     The $STRNLOG system service can be used to obtain the value of
942 1215 1     supervisor mode logical names. The routine is equivalent to the
943 1216 1     DEASSIGN DCL command.
944 1217 1
945 1218 1 CALLING SEQUENCE:
946 1219 1
947 1220 1     ret_status.wlc.v = LIB$DELETE_LOGICAL (logname.rt.dx [,tabnam.rt.dx])
948 1221 1
949 1222 1 FORMAL PARAMETERS:
950 1223 1
951 1224 1     logname.rt.dx - A string, passed by descriptor, which contains the
952 1225 1     supervisor mode logical name to be deleted.
953 1226 1     Must be present.
954 1227 1
955 1228 1     tabnam.rt.dx - A string, passed by descriptor, that contains the
956 1229 1     name of the logical name table from which the
957 1230 1     logical name is to be deleted.
958 1231 1     If omitted, the logical name is removed from
959 1232 1     the LNM$PROCESS logical name table.
960 1233 1
961 1234 1 IMPLICIT INPUTS:
962 1235 1
963 1236 1     It is assumed that the logical name table specified already exists.
964 1237 1
965 1238 1 IMPLICIT OUTPUTS:
966 1239 1
967 1240 1     NONE
968 1241 1
969 1242 1 COMPLETION STATUS: (or ROUTINE VALUE:)
970 1243 1
971 1244 1     $$$_NORMAL      Normal successful completion
972 1245 1     $$$_ACCVIO      logname cannot be read
973 1246 1     $$$_IVLOGNAM     logname contains more than 63 characters
974 1247 1     $$$_NOLOGNAM     No such logical name defined or logical name specified
975 1248 1                     was declared in executative or kernel mode
976 1249 1     $$$_IVLOGTAB     Invalid logical name table specified
977 1250 1     $$$_NOPRIV       Caller lacks the necessary privilege to delete
978 1251 1                     the logical name
979 1252 1     $$$_TOOMANYLNAM  Logical name translation of the table name exceeded
980 1253 1                     the allowable depth (10 translations)
981 1254 1     LIB$_INVARG       Invalid argument
982 1255 1     LIB$_NOCLI        No CLI present to perform function
983 1256 1     LIB$_UNECLIERR    Unexpected CLI Error
984 1257 1     LIB$_WRONUMARG    No arguments specified or more than two specified.

```



```

1258 1
1259 1 SIDE EFFECTS:
1260 1
1261 1 A supervisor mode logical name is deleted from the specified or
1262 1 implied logical name table.
1263 1
1264 1 --
1265 1
1266 1 BEGIN
1267 1
1268 1 BUILTIN
1269 1 ACTUALCOUNT,
1270 1 NULLPARAMETER;
1271 1
1272 1 LOCAL
1273 1 CLI_REQ_BLOCK : BLOCK [CLISC_SRVDESC, BYTE], ! A CLI request block
1274 1 RETURN_STATUS : BLOCK [4, BYTE];
1275 1
1276 1 +
1277 1 Check for proper number of arguments.
1278 1
1279 1
1280 1 IF ACTUALCOUNT() EQLU 0
1281 1 OR ACTUALCOUNT() GTRU 2
1282 1 THEN
1283 1 RETURN LIB$WRONUMARG;
1284 1
1285 1 +
1286 1 Initialize CLI Command request block.
1287 1
1288 1 CH$FILL (0, CLISC_SRVDESC, CLI_REQ_BLOCK);
1289 1 CLI_REQ_BLOCK [CLISB_RQTYPE] = CLISK_CLISERV;
1290 1 CLI_REQ_BLOCK [CLISW_SERVCO] = CLISK_DELELOG;
1291 1
1292 1 +
1293 1 Move in the logical name.
1294 1
1295 1
1296 1 MOVEDESC ( .LOGNAME, CLI_REQ_BLOCK [CLISQ_NAMDESC] );
1297 1
1298 1 +
1299 1 Move in the table name, if specified.
1300 1 If not specified, DCL will default in LNMSPROCESS.
1301 1
1302 1
1303 1 IF NOT NULLPARAMETER(TABNAM)
1304 1 THEN
1305 1 MOVEDESC ( .TABNAM, CLI_REQ_BLOCK [CLISQ_TABDESC] );
1306 1
1307 1 +
1308 1 Delete the supervisor mode logical name.
1309 1
1310 1 RETURN_STATUS = SYS$CLI (CLI_REQ_BLOCK);
1311 1
1312 1 +
1313 1 Adjust error return status, if any.
1314 1
1315 1 IF NOT .RETURN_STATUS AND (.RETURN_STATUS [STSSV_FAC_NO] EQL CLIS_FACILITY)

```

```

1042      1315 2      THEN
1043      1316      RETURN_STATUS =
1044      1317      BEGIN
1045      1318      SELECTONE .RETURN_STATUS OF
1046      1319      SET
1047      1320      [CLIS_INVREQTYP] : LIB$NOCLI;
1048      1321      [OTHERWISE] : LIB$UNECLIERR;
1049      1322      TES
1050      1323      END;
1051      1324
1052      1325      RETURN .RETURN_STATUS
1053      1326
1054      1327      END;

```

! End of LIB\$DELETE_LOGICAL

				007C 00000	.ENTRY LIB\$DELETE LOGICAL, Save R2,R3,R4,R5,R6	1202
	56	00000000G	00	9E 00002	MOVAB LIB\$ANALYZE_SDESC_R2, R6	
	5E	AC	AE	9E 00009	MOVAB -84(SP), SP	
			6C	95 0000D	TSTB (AP)	1280
			05	13 0000F	BEQL 1\$	
	02		6C	91 00011	CMPB (AP), #2	1281
			08	1B 00014	BLEQU 2\$	
	50	00000000G	8F	D0 00016 1\$:	MOVL #LIB\$WRONUMARG, R0	1283
				04 0001D	RET	
0054	BF	00	6E	2C 0001E 2\$:	MOVC5 #0, (SP), #0, #84, CLI_REQ_BLOCK	1287
			6E	00025		
			05	90 00026	MOVAB #5, CLI_REQ_BLOCK	1288
	01		07	B0 00029	MOVW #7, CLI_REQ_BLOCK+1	1289
		04	AC	D0 0002D	MOVL LOGNAME, R0	1295
			66	16 00031	JSB LIB\$ANALYZE_SDESC_R2	
	04		51	B0 00033	MOVW R1, CLI_REQ_BLOCK+4	
	08		52	D0 00037	MOVL R2, CLI_REQ_BLOCK+8	
			50	E9 0003B	BLBC RET_STATUS, 5\$	
			6C	91 0003E	CMPB (APT, #2	1302
			16	1F 00041	BLSSU 3\$	
		08	AC	D5 00043	TSTL 8(AP)	
			11	13 00046	BEQL 3\$	
		50	AC	D0 00048	MOVL TABNAM, R0	1304
			66	16 0004C	JSB LIB\$ANALYZE_SDESC_R2	
	14		51	B0 0004E	MOVW R1, CLI_REQ_BLOCK+20	
	18		52	D0 00052	MOVL R2, CLI_REQ_BLOCK+24	
			50	E9 00056	BLBC RET_STATUS, 5\$	
			5E	DD 00059 3\$:	PUSHL SP	1309
		00000000G	00	01 FB 0005B	CALLS #1, SYSSCLI	
			1F	50 E8 00062	BLBS RETURN_STATUS, 5\$	1314
03		50	0C	1C ED 00065	CMPZV #16, #T2, RETURN_STATUS, #3	
			18	12 0006A	BNEQ 5\$	
		00038822	8F	50 D1 0006C	CMPB RETURN_STATUS, #231458	1320
			08	12 00073	BNEQ 4\$	
			50	00000000G 8F D0 00075	MOVL #LIB\$NOCLI, RETURN_STATUS	
				04 0007C	RET	
			50	00000000G 8F D0 0007D 4\$:	MOVL #LIB\$UNECLIERR, RETURN_STATUS	1321
				04 00084 5\$:	RET	1327

LIB\$CLI_CALLBAC LIB\$CLI CALLBACK - CLI Callback Interface Proce 6 10
V04-000 LIB\$DELETE_LOGICAL - delete supervisor mode log 16-Sep-1984 02:22:35
14-Sep-1984 13:27:47

VAX-11 Bliss-32 V4.0-742
[VMSLIB.SRC]LIBCLICAL.B32;1

Page 29
(7)

; Routine Size: 133 bytes, Routine Base: _LIB\$CODE + 037B


```

1056 1328 1 %SBTTL 'LIB$DISABLE_CTRL - disable CLI out-of-band character(s) handler'
1057 1329 1 GLOBAL ROUTINE LIB$DISABLE_CTRL (
1058 1330 1     DISABLE_MASK : REF VECTOR [ , LONG]      ! disable CLI out-of-band
1059 1331 1     PREVIOUS_MASK : REF VECTOR [ , LONG]      ! disable bit mask
1060 1332 1 ) =
1061 1333 1
1062 1334 1 ++
1063 1335 1 FUNCTIONAL DESCRIPTION:
1064 1336 1
1065 1337 1     This routine disables CLI processing for out-of-band the characters
1066 1338 1     identified by DISABLE_MASK. When normal CLI processing for one or
1067 1339 1     more out-of-band characters is disabled, a user program can intercept
1068 1340 1     and respond to them. There are no privilege requirements for calling
1069 1341 1     this routine.
1070 1342 1
1071 1343 1     Currently, this routine supports disabling of CLI out-of-band
1072 1344 1     processing for CTRL-T and CTRL-Y when the DCL CLI is in use, and for
1073 1345 1     CTRL-Y when the MCR CLI is in use. The macro $LIBCLIDEF defines the
1074 1346 1     two masks used for disabling processing for these two out-of-band
1075 1347 1     characters as follows:
1076 1348 1         LIB$M_CLI_CTRLT      mask for CTRL-T
1077 1349 1         LIB$M_CLI_CTRLY      mask for CTRL-Y
1078 1350 1     If the masks are logically or'd together, processing for both
1079 1351 1     out-of-band characters will be effected.
1080 1352 1
1081 1353 1 CALLING SEQUENCE:
1082 1354 1
1083 1355 1     ret_status.wlc.v = LIB$DISABLE_CTRL (disable_mask.rl.r
1084 1356 1                                     [ , previous_mask.wl.r ] )
1085 1357 1
1086 1358 1 FORMAL PARAMETERS:
1087 1359 1
1088 1360 1     disable_mask.rl.r      - the mask controlling which out-of-band
1089 1361 1                             characters the CLI should ignore. Values
1090 1362 1                             should be selected from one of, or the logical
1091 1363 1                             OR of more than one of:
1092 1364 1                                 LIB$M_CLI_CTRLT
1093 1365 1                                 LIB$M_CLI_CTRLY
1094 1366 1                             These symbols are defined in $LIBCLIDEF.
1095 1367 1
1096 1368 1     previous_mask.wl.r     - a longword into which will be stored a mask
1097 1369 1                             having the same format as disable_mask, but
1098 1370 1                             which represents the enabled state for all
1099 1371 1                             out-of-band character processing before
1100 1372 1                             disable_mask is applied.
1101 1373 1
1102 1374 1 IMPLICIT INPUTS:
1103 1375 1
1104 1376 1     NONE
1105 1377 1
1106 1378 1 IMPLICIT OUTPUTS:
1107 1379 1
1108 1380 1     NONE
1109 1381 1
1110 1382 1 COMPLETION STATUS: (or ROUTINE VALUE:)
1111 1383 1
1112 1384 1     $$$_NORMAL      Normal successful completion
  
```

```

1113 1385 1 LIB$ INVARG Disable_mask has an unrecognized bit set
1114 1386 1 LIB$ NOCLI No CLI present to perform function
1115 1387 1 LIB$ UNECLIERR Unexpected CLI Error
1116 1388
1117 1389 SIDE EFFECTS:
1118 1390
1119 1391 CLI out-of-band processing of one or more characters is disabled.
1120 1392
1121 1393
1122 1394
1123 1395 BEGIN
1124 1396
1125 1397 BUILTIN
1126 1398 NULLPARAMETER;
1127 1399
1128 1400 LOCAL
1129 1401 CLI_REQ_BLOCK : BLOCK [CLIS$ SRVDESC, BYTE], ! A CLI request block
1130 1402 RETURN_STATUS : BLOCK [4, BYTE];
1131 1403
1132 1404
1133 1405 +
1134 1406 Initialize CLI Command request block.
1135 1407
1136 1408 CH$FILL (0, CLIS$ SRVDESC, CLI_REQ_BLOCK);
1137 1409 CLI_REQ_BLOCK [CLIS$ RQTYPE] = CLIS$ CLISERV;
1138 1410 CLI_REQ_BLOCK [CLIS$ SERVCOD] = CLIS$ DISAOOB;
1139 1411 CLI_REQ_BLOCK [CLIS$ NEW_MASK] = .DISABLE_MASK [0];
1140 1412
1141 1413 +
1142 1414 Request CLI out-of-band processing be disabled.
1143 1415
1144 1416 RETURN_STATUS = SYS$CLI (CLI_REQ_BLOCK);
1145 1417
1146 1418 +
1147 1419 If requested, return previous out-of-band enabled mask.
1148 1420 It may be good.
1149 1421
1150 1422 IF NOT NULLPARAMETER (2)
1151 1423 THEN
1152 1424 PREVIOUS_MASK [0] = .CLI_REQ_BLOCK [CLIS$ OLD_MASK];
1153 1425
1154 1426 +
1155 1427 Adjust error return status, if any.
1156 1428
1157 1429 IF NOT .RETURN_STATUS AND (.RETURN_STATUS [STSSV_FAC_NO] EQL CLIS$ FACILITY)
1158 1430 THEN
1159 1431 RETURN_STATUS =
1160 1432 BEGIN
1161 1433 SELECTONE .RETURN_STATUS OF
1162 1434 SET
1163 1435 [CLIS$ BADCTLMSK] : LIB$ INVARG;
1164 1436 [CLIS$ INVREQTYP] : LIB$ NOCLI;
1165 1437 [OTHERWISE] : LIB$ UNECLIERR;
1166 1438 YES
1167 1439 END;
1168 1440
1169 1441 RETURN (.RETURN_STATUS);

```

: 1170 1442 2
: 1171 1443 1 END;

! End of LIB\$DISABLE_CTRL

0054	8F	00	5E	AC	AE	003C	00000	.ENTRY	LIB\$DISABLE_CTRL, Save R2,R3,R4,R5	1329
			6E		00	9E	00002	MOVAB	-84(SP), SP	
					00	2C	00006	MOVC5	#0, (SP), #0, #84, CLI_REQ_BLOCK	1408
					6E		0000D			
		01	6E		05	90	0000E	MOVAB	#5, CLI_REQ_BLOCK	1409
		04	AE		0D	B0	00011	MOVW	#13, CLI_REQ_BLOCK+1	1410
			AE	04	BC	D0	00015	MOVL	@DISABLE_MASR, CLI_REQ_BLOCK+4	1411
					5E	DD	0001A	PUSHL	SP	1416
		00000000G	00		01	FB	0001C	CALLS	#1, SYSSCLI	
			02		6C	91	00023	CMPB	(AP), #2	1422
					0A	1F	00026	BLSSU	1\$	
				08	AC	D5	00028	TSTL	8(AP)	
		08	BC	08	05	13	0002B	BEQL	1\$	
			30		AE	D0	0002D	MOVL	CLI_REQ_BLOCK+8, @PREVIOUS_MASK	1424
03		50	0C		50	E8	00032	BLBS	RETURN_STATUS, 4\$	1429
					10	ED	00035	CMPZV	#16, #12, RETURN_STATUS, #3	
		000388CA	8F		29	12	0003A	BNEQ	4\$	
					50	D1	0003C	CML	RETURN_STATUS, #231626	1435
			50	00000000G	08	12	00043	BNEQ	2\$	
					8F	D0	00045	MOVL	#LIB\$_INVARG, RETURN_STATUS	
		00038822	8F		04	0004C		RET		
					50	D1	0004D	CML	RETURN_STATUS, #231458	1436
			50	00000000G	08	12	00054	BNEQ	3\$	
					8F	D0	00056	MOVL	#LIB\$_NOCLI, RETURN_STATUS	
					04	0005D		RET		
		50	00000000G		8F	D0	0005E	MOVL	#LIB\$_UNECLIERR, RETURN_STATUS	1437
					04	00065		RET		1443

; Routine Size: 102 bytes, Routine Base: _LIB\$CODE + 0400


```

1173 1444 1 %SBTTL 'LIB$ENABLE_CTRL - re-enable CLI out-of-band character(s) handler'
1174 1445 1 GLOBAL ROUTINE LIB$ENABLE_CTRL ( re-enable CLI out-of-band
1175 1446 1 ENABLE_MASK : REF VECTOR [, LONG], enable bit mask
1176 1447 1 PREVIOUS_MASK : REF VECTOR [, LONG] previous enable bit mask
1177 1448 1 ) =
1178 1449 1
1179 1450 1 ++
1180 1451 1 FUNCTIONAL DESCRIPTION:
1181 1452 1
1182 1453 1 This routine re-enables CLI processing for out-of-band the characters
1183 1454 1 identified by ENABLE_MASK, presumable after such processing has been
1184 1455 1 disabled by a call to LIB$DISABLE_CTRL. There are no privilege
1185 1456 1 requirements for calling this routine.
1186 1457 1
1187 1458 1 Currently, this routine supports re-enabling of CLI out-of-band
1188 1459 1 processing for CTRL-T and CTRL-Y when the DCL CLI is in use, and for
1189 1460 1 CTRL-Y when the MCR CLI is in use. The macro $LIBCLIDEF defines the
1190 1461 1 two masks used for enabling processing for these two out-of-band
1191 1462 1 characters as follows:
1192 1463 1 LIB$M_CLI_CTRLT mask for CTRL-T
1193 1464 1 LIB$M_CLI_CTRLY mask for CTRL-Y
1194 1465 1 If the masks are logically or'd together, processing for both
1195 1466 1 out-of-band characters will be effected.
1196 1467 1
1197 1468 1 CALLING SEQUENCE:
1198 1469 1
1199 1470 1 ret_status.wlc.v = LIB$ENABLE_CTRL (enable_mask.rl.r
1200 1471 1 [, previous_mask.wl.r ] )
1201 1472 1
1202 1473 1 FORMAL PARAMETERS:
1203 1474 1
1204 1475 1 enable_mask.rl.r - the mask controlling which out-of-band
1205 1476 1 characters the CLI should process. Values
1206 1477 1 should be selected from one of, or the logical
1207 1478 1 OR of more than one of:
1208 1479 1 LIB$M_CLI_CTRLT
1209 1480 1 LIB$M_CLI_CTRLY
1210 1481 1 These symbols are defined in $LIBCLIDEF.
1211 1482 1
1212 1483 1 previous_mask.wl.r - a longword into which will be stored a mask
1213 1484 1 having the same format as enable_mask, but
1214 1485 1 which represents the enabled state for all
1215 1486 1 out-of-band character processing before
1216 1487 1 enable_mask is applied.
1217 1488 1
1218 1489 1 IMPLICIT INPUTS:
1219 1490 1
1220 1491 1 NONE
1221 1492 1
1222 1493 1 IMPLICIT OUTPUTS:
1223 1494 1
1224 1495 1 NONE
1225 1496 1
1226 1497 1 COMPLETION STATUS: (or ROUTINE VALUE:)
1227 1498 1
1228 1499 1 $$$ NORMAL Normal successful completion
1229 1500 1 LIB$INVARG Enable_mask has an unrecognized bit set

```

```
1230 1501 1 LIB$NOCLI No CLI present to perform function
1231 1502 1 LIB$UNECLIERR Unexpected CLI Error
1232 1503 1
1233 1504 1 SIDE EFFECTS:
1234 1505 1
1235 1506 1 CLI out-of-band processing of one or more characters is enabled.
1236 1507 1
1237 1508 1
1238 1509 1
1239 1510 2 BEGIN
1240 1511 2
1241 1512 2 BUILTIN
1242 1513 2 NULLPARAMETER;
1243 1514 2
1244 1515 2 LOCAL
1245 1516 2 CLI_REQ_BLOCK : BLOCK [CLISC_SRVDDESC, BYTE], ! A CLI request block
1246 1517 2 RETURN_STATUS : BLOCK [4, BYTE];
1247 1518 2
1248 1519 2
1249 1520 2 + Initialize CLI Command request block.
1250 1521 2
1251 1522 2
1252 1523 2 CH$FILL (0, CLISC_SRVDDESC, CLI_REQ_BLOCK);
1253 1524 2 CLI_REQ_BLOCK [CLISB_RQTYPE] = CLISK_CLISERV;
1254 1525 2 CLI_REQ_BLOCK [CLISW_SERVCOB] = CLISK_ENABOOB;
1255 1526 2 CLI_REQ_BLOCK [CLISL_NEW_MASK] = .ENABLE_MASK [0];
1256 1527 2
1257 1528 2 +
1258 1529 2 Request CLI out-of-band processing be enabled.
1259 1530 2
1260 1531 2 RETURN_STATUS = SYS$CLI (CLI_REQ_BLOCK);
1261 1532 2
1262 1533 2 +
1263 1534 2 If requested, return previous out-of-band enabled mask.
1264 1535 2 It may be good.
1265 1536 2
1266 1537 2 IF NOT NULLPARAMETER (2)
1267 1538 2 THEN
1268 1539 2 PREVIOUS_MASK [0] = .CLI_REQ_BLOCK [CLISL_OLD_MASK];
1269 1540 2
1270 1541 2 +
1271 1542 2 Adjust error return status, if any.
1272 1543 2
1273 1544 2 IF NOT .RETURN_STATUS AND (.RETURN_STATUS [STSSV_FAC_NO] EQL CLIS_FACILITY)
1274 1545 2 THEN
1275 1546 2 RETURN_STATUS =
1276 1547 2 BEGIN
1277 1548 2 SELECTONE .RETURN_STATUS OF
1278 1549 2 SET
1279 1550 2 [CLIS_BADCTLMSK] : LIB$INVARG;
1280 1551 2 [CLIS_INVREQTYP] : LIB$NOCLI;
1281 1552 2 [OTHERWISE] : LIB$UNECLIERR;
1282 1553 2 YES
1283 1554 2 END;
1284 1555 2
1285 1556 2 RETURN (.RETURN_STATUS);
1286 1557 2
```

0054	8F	00	5E	AC	AE	9E	00002	.ENTRY	LIB\$ENABLE_CTRL, Save R2,R3,R4,R5	1445
			6E		00	2C	00006	MOVAB	-84(SP), SP	
					6E		0000D	MOVCS	#0, (SP), #0, #84, CLI_REQ_BLOCK	1523
		01	6E		05	90	0000E	MOVB	#5, CLI_REQ_BLOCK	1524
		04	AE	04	0E	B0	00011	MOVW	#14, CLI_REQ_BLOCK+1	1525
					BC	D0	00015	MOVL	@ENABLE_MASK, CLI_REQ_BLOCK+4	1526
		00000000G	00		5E	DD	0001A	PUSHL	SP	1531
			02		01	FB	0001C	CALLS	#1, SYS\$CLI	
					6C	91	00023	CMPB	(AP), #2	1537
				08	0A	1F	00026	BLSSU	1\$	
					AC	D5	00028	TSTL	8(AP)	
		08	BC	08	05	13	0002B	BEQL	1\$	
			30		AE	D0	0002D	MOVL	CLI_REQ_BLOCK+8, @PREVIOUS_MASK	1539
03		50	0C		50	E8	00032	BLBS	RETURN_STATUS, 4\$	1544
					10	ED	00035	CMPZV	#16, #T2, RETURN_STATUS, #3	
		000388CA	8F		29	12	0003A	BNEQ	4\$	
					50	D1	0003C	CML	RETURN_STATUS, #231626	1550
			50 00000000G		08	12	00043	BNEQ	2\$	
					8F	D0	00045	MOVL	#LIB\$_INVARG, RETURN_STATUS	
		00038822	8F			04	0004C	RET		
					50	D1	0004D	CML	RETURN_STATUS, #231458	1551
			50 00000000G		08	12	00054	BNEQ	3\$	
					8F	D0	00056	MOVL	#LIB\$_NOCLI, RETURN_STATUS	
						04	0005D	RET		
			50 00000000G		8F	D0	0005E	MOVL	#LIB\$_UNECLIERR, RETURN_STATUS	1552
					04	00065	4\$:	RET		1558

; Routine Size: 102 bytes. Routine Base: _LIB\$CODE + 0466


```

1289 1559 1 %SBTTL 'LIB$BUILD_SYMBOL_NAME - Build a symbol name string'
1290 1560 1 ROUTINE LIB$BUILD_SYMBOL_NAME (
1291 1561 1     SYMBOL: REF BLOCK [, BYTE],      ! build a symbol name string
1292 1562 1     DYNSTR: REF BLOCK [, BYTE],      ! input symbol string descriptor
1293 1563 1     CLI_SYMBOL: REF BLOCK [, BYTE] ! dynamic string descriptor
1294 1564 1 ) =                               ! CLI symbol string descriptor
1295 1565 1
1296 1566 1 ++
1297 1567 1 FUNCTIONAL DESCRIPTION:
1298 1568 1
1299 1569 1     Using the string pointed to by SYMBOL, LIB$BUILD_SYMBOL_NAME builds a
1300 1570 1     dynamic string, pointed to by CLI_SYMBOL, which has been upcased and
1301 1571 1     has trailing blanks deleted. The string pointed to by SYMBOL is also
1302 1572 1     tested for validity as a symbol name; the first character must be
1303 1573 1     alphabetic or $ or _ there must be less than 255 characters in the name
1304 1574 1     (trailing blanks excluded), and there must be at least one non-blank
1305 1575 1     character in the name.
1306 1576 1
1307 1577 1     SCRATCH is a string descriptor into which the actual descriptor for
1308 1578 1     the dynamic string will be written. The calling procedure must use
1309 1579 1     this descriptor to deallocate the dynamic string.
1310 1580 1
1311 1581 1     If this routine detects an error after it has allocated the dynamic
1312 1582 1     string, it will deallocate the dynamic string thus freeing its caller
1313 1583 1     from that responsibility.
1314 1584 1
1315 1585 1 CALLING SEQUENCE:
1316 1586 1
1317 1587 1     ret_status.wlc.v = LIB$BUILD_SYMBOL_NAME (symbol.rt.dx,
1318 1588 1                                           dynstr.wt.dx,
1319 1589 1                                           cli_symbol.wt.dx)
1320 1590 1
1321 1591 1 FORMAL PARAMETERS:
1322 1592 1
1323 1593 1     symbol.rt.dx - A string, passed by descriptor, which contains the
1324 1594 1                   original symbol name.
1325 1595 1
1326 1596 1     dynstr.wt.dx - A string descriptor into which the descriptor for the
1327 1597 1                   actual dynamic string used to contain the converted
1328 1598 1                   symbol name will be written. The calling procedure
1329 1599 1                   must use this descriptor to deallocate the dynamic
1330 1600 1                   string.
1331 1601 1
1332 1602 1     cli_symbol.wt.dx - A string, passed by descriptor, into which the
1333 1603 1                       processed symbol name will be placed. This routine
1334 1604 1                       creates a new dynamic string in this descriptor; the
1335 1605 1                       contents of the descriptor upon entry are overwritten.
1336 1606 1
1337 1607 1 IMPLICIT INPUTS:
1338 1608 1
1339 1609 1     LIB$AB_UPCASE the address of an upcase translation table.
1340 1610 1
1341 1611 1 IMPLICIT OUTPUTS:
1342 1612 1
1343 1613 1     NONE
1344 1614 1
1345 1615 1 COMPLETION STATUS: (or ROUTINE VALUE:)

```

```
1346 1616 1 |
1347 1617 1 |      SSS NORMAL      Normal successful completion
1348 1618 1 |      LIB$INVARG      Symbol name contains more than 255 characters or is
1349 1619 1 |                      all blanks
1350 1620 1 |      LIB$INVSYMNAM    Symbol name does not begin with a letter
1351 1621 1 |      LIB$INSVIRMEM    Insufficient virtual memory for dynamic string
1352 1622 1 |      Other completion stati from LIB$ANALYZE_SDESC.
1353 1623 1 |
1354 1624 1 |
1355 1625 1 | SIDE EFFECTS:
1356 1626 1 |
1357 1627 1 | WARNING:
1358 1628 1 | Although this procedure performs some checks on the validity of
1359 1629 1 | symbol names passed to it, some symbol name considered valid by
1360 1630 1 | this procedure will be considered invalid by DCL.
1361 1631 1 |
1362 1632 1 | --
1363 1633 1 |
1364 1634 1 | BEGIN
1365 1635 1 |
1366 1636 1 | LOCAL
1367 1637 1 |     ADDRESS,          | a string address
1368 1638 1 |     LENGTH : WORD,    | a string length
1369 1639 1 |     IN_PTR, OUT_PTR,  | two pointer variables
1370 1640 1 |     STATUS;           | a return status value
1371 1641 1 |
1372 1642 1 |
1373 1643 1 |
1374 1644 1 | +
1375 1645 1 | Gather information about input symbol name string
1376 1646 1 | and build pointer input symbol name string.
1377 1647 1 | -
1378 1648 1 |
1379 1649 1 | STATUS = LIB$ANALYZE_SDESC_R2 (.SYMBOL; LENGTH, ADDRESS);
1380 1650 1 | IF NOT .STATUS THEN RETURN .STATUS;
1381 1651 1 | IN_PTR = CH$PTR( .ADDRESS, .LENGTH - 1 );
1382 1652 1 |
1383 1653 1 | +
1384 1654 1 | Work backwards from the end of the string to locate
1385 1655 1 | the first non-blank character; i.e. locate beginning
1386 1656 1 | of the trailing blanks.
1387 1657 1 | -
1388 1658 1 | LENGTH = 1 +
1389 1659 1 |     BEGIN
1390 1660 1 |         DECR J FROM .LENGTH - 1 TO 0 DO
1391 1661 1 |         BEGIN
1392 1662 1 |             IF CH$RCHAR( .IN_PTR ) NEQU %C' ' THEN EXITLOOP .J;
1393 1663 1 |             IN_PTR = CH$PLUST .IN_PTR, -1 );
1394 1664 1 |         END
1395 1665 1 |     END;
1396 1666 1 | IF .LENGTH EQLU 0 OR .LENGTH GTRU 255 THEN RETURN LIB$INVSYMNAM;
1397 1667 1 |
1398 1668 1 | +
1399 1669 1 | Allocate a dynamic string and setup CLI symbol name
1400 1670 1 | string descriptor.
1401 1671 1 | -
1402 1672 1 | DYNSTR [DSC$B_CLASS] = DSC$K_CLASS_D;
```

```

1403 1673 2 DYNSTR [DSC$B_DTYPE] = DSC$K_DTYPE_T;
1404 1674 2 DYNSTR [DSC$W_LENGTH] = 0;
1405 1675 2 DYNSTR [DSC$A_POINTER] = 0;
1406 1676 2 STATUS = LIB$GET1 DD( LENGTH, DYNSTR );
1407 1677 2 IF NOT .STATUS THEN RETURN .STATUS;
1408 1678 2 CLI_SYMBOL [DSC$A_POINTER] = .DYNSTR [DSC$A_POINTER];
1409 1679 2 CLI_SYMBOL [DSC$W_LENGTH] = .LENGTH;
1410 1680 2
1411 1681 2 +
1412 1682 2 Upcase the symbol name.
1413 1683 2 -
1414 1684 2 IN_PTR = CH$PTR( .ADDRESS );
1415 1685 2 OUT_PTR = CH$PTR( .CLI_SYMBOL [DSC$A_POINTER] );
1416 1686 2 CH$TRANSLATE( LIB$AB_UPCASE, .LENGTH, .IN_PTR, %C' ', .LENGTH, .OUT_PTR );
1417 1687 2
1418 1688 2 +
1419 1689 2 Check for reasonable first character in symbol name.
1420 1690 2 -
1421 1691 2 IF (CH$RCHAR( .OUT_PTR ) LSSU %C'A' OR CH$RCHAR( .OUT_PTR ) GTRU %C'Z') AND
1422 1692 2 (CH$RCHAR( .OUT_PTR ) NEQU %C'$' OR CH$RCHAR( .OUT_PTR ) NEQU %C'_')
1423 1693 2 THEN
1424 1694 2 BEGIN
1425 1695 2 LIB$FREE1 DD( .DYNSTR );
1426 1696 2 RETURN LIB$_INVSYMNAM;
1427 1697 2 END
1428 1698 2 ELSE
1429 1699 2 RETURN SSS_NORMAL;
1430 1700 2
1431 1701 1 END;

```

! End of routine LIB\$\$BUILD_SYMBOL_NAME

```

00FC 00000 LIB$$BUILD SYMBOL NAME:
5E          04 04 C2 00002      .WORD      Save R2,R3,R4,R5,R6,R7      : 1560
50          00 AC D0 00005      SUBL2      #4, SP                      : 1649
000000000G 00 16 00009      JSB          LIB$ANALYZE_SDESC_R2
53          52 D0 0000F      MOVL        R2, R3
6E          51 D0 00012      MOVL        R1, LENGTH
41          50 E9 00015      BLBC        STATUS, 4$                    : 1650
52          6E 3C 00018      MOVZWL     LENGTH, R2                    : 1651
52          53 C0 0001B      ADDL2      ADDRESS, R2
51          52 D7 0001E      DECL        IN_PTR
20          6E 3C 00020      MOVZWL     LENGTH, J                      : 1660
07          11 00023      BRB          2$
08          62 91 00025 1$:    CMPB      (IN_PTR), #32                  : 1662
08          12 00028      BNEQ        3$
52          D7 0002A      DECL        IN_PTR
F6          51 F4 0002C 2$:    SOBGEQ   J, -1$                          : 1663
51          01 CE 0002F      MNEGL     #1, R1                          : 1660
6E          51 01 A1 00032 3$:    ADDW3   #1, R1, LENGTH                    : 1658
00FF 8F      62 13 00036      BEQL      7$                          : 1666
56          5B 1A 0003D      BGTRU     7$
08          AC D0 0003F      MOVL        DYNSTR, R6                      : 1672

```


	66	020E0000	8F	D0	00043	MOVL	#34471936, (R6)	:	1674
			04	A6	D4	CLRL	4(R6)	:	1675
				56	DD	PUSHL	R6	:	1676
			04	AE	9F	PUSHAB	LENGTH	:	
00000000G	00			02	FB	CALLS	#2, LIB\$\$GET1_DD	:	
	49			50	E9	BLBC	STATUS, 9\$:	1677
	50	0C		AC	D0	MOVL	CLI_SYMBOL, R0	:	1678
	04		04	A6	D0	MOVL	4(R6), 4(R0)	:	
	60			6E	B0	MOVW	LENGTH, (R0)	:	1679
	52			53	D0	MOVL	ADDRESS, IN_PTR	:	1684
	57		04	A0	D0	MOVL	4(R0), OUT_PTR	:	1685
00000000G	00	20		62	2E	MOVTC	LENGTH, (IN_PTR), #32, LIB\$AB_UPCASE, -	:	1686
	67			6E			LENGTH, (OUT_PTR)	:	
	41			67	91	CMPB	(OUT_PTR), #85	:	1691
	8F			06	1F	BLSSU	5\$:	
	5A			67	91	CMPB	(OUT_PTR), #90	:	
				1C	1B	BLEQU	8\$:	
	24			67	91	CMPB	(OUT_PTR), #36	:	1692
				06	12	BNEQ	6\$:	
	5F			67	91	CMPB	(OUT_PTR), #95	:	
				11	13	BEQL	8\$:	
				56	DD	PUSHL	R6	:	1695
00000000G	00			01	FB	CALLS	#1, LIB\$\$FREE1_DD	:	
	50	00000000G		8F	D0	MOVL	#LIB\$_INVSYMNAM, R0	:	1699
				04	00A1	RET		:	
	50			01	D0	MOVL	#1, R0	:	
				04	00A5	RET		:	1701

; Routine Size: 166 bytes, Routine Base: _LIB\$CODE + 04CC

LIB\$CLI_CALLBAC LIB\$CLI_CALLBACK - CLI Callback Interface Proce

VO4-000 LIB\$\$BUILD_SYMBOL_NAME - Build a symbol name st

E 11

16-Sep-1984 02:22:35

VAX-11 Bliss-32 V4.0-742

Page 40

14-Sep-1984 13:27:47

[VMSLIB.SRC]LIBCLICAL.B32;1

(11)

: 1433

: 1434

1702 1 END

1703 0 ELUDOM

! End of module LIB\$CLI_CALLBACK

: PSECT SUMMARY

: Name Bytes Attributes

: _LIB\$CODE 1394 NOVEC,NOWRT, RD , EXE, SHR, LCL, REL, CON, PIC,ALIGN(2)

: Library Statistics

: File Total Symbols Loaded Percent Pages Mapped Processing Time

: _\$255\$DUA28:[SYSLIB]STARLET.L32;1 9776 38 0 581 00:01.0

: COMMAND QUALIFIERS

: BLISS/CHECK=(FIELD,INITIAL,OPTIMIZE)/NOTRACE/LIS=LISS:LIBCLICAL/OBJ=OBJ\$:LIBCLICAL MSRC\$:LIBCLICAL/UPDATE=(ENHS:LIBCLICAL

:)

: Size: 1394 code + 0 data bytes

: Run Time: 00:34.2

: Elapsed Time: 01:07.4

: Lines/CPU Min: 2990

: Lexemes/CPU-Min: 37106

: Memory Used: 171 pages

: Compilation Complete

DIGITAL EQUIPMENT CORPORATION
CONFIDENTIAL AND PROPRIETARY